**Research for other solutions**

**1, Kaggle competition 1st place solution**

**1) Pipeline**

1. **Tabular feature engineering**

Build a compact **4-D feature vector** per patient entry (normalized Age, one-hots/binaries for Sex, SmokingStatus, etc.).  
Result: for each row in train.csv can get a (4,) numeric vector.

**Per-patient linear trend estimation (baseline):**   
For each patient, fit a simple **least-squares line** FVC = a \* Weeks + b. Store slope a (rate of weekly FVC change) and per-patient tabular vector for downstream use.

Purpose: a strong baseline and helpful prior/trend structure.

1. **Image preprocessing (CTs)**

Read DICOM slices with pydicom and cv2.

Window/level & intensity normalization (typical Hounsfield preprocessing), select central/representative slices or form a stack.

Resize to the CNN input shape (e.g., (512, 512, 1)).

1. **Two modeling routes**

# Route A — Hybrid CNN + Tabular (EfficientNet-B5 backbone)

## What problem it solves

Uses **CT images** and **patient tabular features** together to predict a **single FVC value** (point estimate). Images carry disease texture/pattern signals; tabular data adds context (age/sex/smoking).

## Inputs

* **Image:** one preprocessed CT slice/stack resized to **(512, 512, 1)** (grayscale).
* **Tabular:** a **4-number vector** built earlier (normalized age + encoded sex + encoded smoking status).

## Architecture (step by step)

1. **Image branch (feature extractor)**
   * EfficientNetB5(include\_top=False, weights=None) → removes ImageNet classifier head.
   * Pass the image through the EfficientNet **convolutional body** → a deep **feature map**.
   * GlobalAveragePooling2D() → collapses the feature map to a **1-D vector** (the **image embedding**).
2. **Tabular branch**
   * Input(shape=(4,)) → the 4-D vector.
   * GaussianNoise(0.2) → adds tiny noise at train time to reduce overfitting (regularization).
3. **Fusion + head**
   * Concatenate([image\_embedding, tab\_vector]) → a single fused vector.
   * Dropout(0.5) → more regularization.
   * Dense(1) → outputs **one number**: the predicted **FVC**.
4. **Weights**
   * The notebook **loads a trained checkpoint**: effnet\_30.h5.  
     This means EfficientNet is already trained to extract useful radiology features; you use it essentially as a **feature extractor** plus a light regression head.

## Training & evaluation

* In **this** notebook, you **don’t re-train** the hybrid model; you load pre-trained weights and infer.
* If you wanted to train/fine-tune:
  + Start with the EfficientNet **frozen** (base.trainable=False), train the fusion head.
  + Optionally unfreeze the **top blocks** for gentle fine-tuning with a small learning rate.
* **Metric:** You can evaluate this route with MAE or compare its predictions to the tabular interval route’s median.

## Output

* **One value per row**: FVC\_pred (point estimate).

## Strengths & caveats

* **Strengths:** Uses image texture (ground-glass, reticulations, distribution) + patient context → often the best raw accuracy.
* **Caveats:** Needs good image preprocessing; heavier compute; and in this notebook it’s used with pre-trained weights (training from scratch needs care and time).

# Route B — Tabular-only Interval/Quantile Head (ordered lower / middle / upper)

## What problem it solves

Predicts **three ordered values** (lower, middle, upper) from **tabular data only**, giving both a **point prediction** (middle) and an **uncertainty band** (upper−lower). This matches the OSIC scoring style that rewards accurate and **well-calibrated** uncertainty.

## Inputs

* **Tabular:** the same **4-number vector** (age/sex/smoking features).

## Architecture (step by step)

1. **Two hidden layers**
   * Dense(100, relu) → Dense(100, relu) to learn a nonlinear summary of the tabular features.
2. **Split into base + increments**
   * p1 = Dense(3, linear) → any real numbers (no constraint).
   * p2 = Dense(3, relu) → **non-negative** increments.
3. **Enforce ordered outputs**
   * preds = p1 + cumsum(p2, axis=1) → guarantees  
     lower = preds[:,0] ≤ middle = preds[:,1] ≤ upper = preds[:,2]  
     because cumulative sums of non-negative numbers can only go **up**.

## Metric (OSIC-style Laplace proxy)

* **Point prediction**: fvc\_pred = middle.
* **Uncertainty**: sigma = upper − lower, but clipped to at least **70** (avoid tiny, over-confident sigmas).
* **Clipped error**: delta = |true − fvc\_pred|, but capped at **1000** (avoid runaway penalties).
* **Score (lower is better)**:

A math symbols with black text

AI-generated content may be incorrect.

Intuition: if you’re wrong but admit it (big sigma), penalty softens; but always predicting huge sigma is punished by the log term.

## Loss (training objective)

* A **quantile/pinball loss** (via mloss(...) in your code) encourages the three outputs to behave like **lower/median/upper quantiles** rather than arbitrary numbers.

## Output

* **Three numbers per row:** (lower, middle, upper).
  + Use **middle** as the **point** FVC.
  + Use sigma = upper − lower to derive **Confidence** for submission.

## Strengths & caveats

* **Strengths:** Fast and simple; gives **uncertainty**; no image preprocessing required; aligns directly with the competition metric.
* **Caveats:** Ignores imaging, so it may miss disease texture/severity signals present in CTs; accuracy can be lower than the hybrid route on image-rich tasks.

1. **Metric & loss (OSIC-style)**
2. **Evaluation & (optional) CV**
   * Train/validate splits or K-Fold (common in Kaggle notebooks).
   * Monitor score and MAE to judge fit & calibration.
3. **Inference & submission**
   * For each target row in the test/submit template, produce:
     + **FVC** (from hybrid or median of tabular head),
     + **Confidence** (linked to sigma, constructed to match the competition format).
   * Write out submission.csv.

**What didn't work**

As I have said I have tried a lot of stuff, but it almost always worked badly both on LB and CV. Here are a few things:

* Calculated lung volume with methods from the public notebooks and passed it as features for both models
* Tested other models, XGBoost, Log Regressions on tabular data. Thanks to my CV it immediately turned out that trees do not work here, so I didn't do anything with trees since the beginning of this competition.
* Since I was testing simple models, my 2nd selected submission was a really simple logistic regression model, which by the way landed in the bronze zone
* Augmentations for the CT scans worked bad, maybe I should have spent more time testing them
* Histogram features of the image didn't work either
* If you have analyzed model outputs, you might have noticed those spikes (both for Confidence and FVC)

**2, One article approach (OSIC Pulmonary Fibrosis Progression)**

<https://medium.com/swlh/osic-pulmonary-fibrosis-progression-5f06febebe0>

**1, DICOM handling & lung segmentation**

* **Metadata extraction:** Uses **pydicom** to read each scan’s header (e.g., pixel spacing, slice thickness, etc.). A custom function iterates and extracts structured meta info for analysis.
* **Segmentation pipeline:** Adapts a Kaggle notebook to segment lung fields so the CNN focuses on lung tissue. Steps: image normalization → clustering for lung vs non-lung → thresholding → morphology (erosion/dilation) → connected-component labeling → build lung mask → apply mask to original scan.

**2, Data preparation challenges (why a custom approach)**

There are **two mismatches**:

1. The number of **CT slices** per patient varies widely.
2. The number of **weekly FVC records** per patient also varies.  
   This makes a single, uniform “one image + one row” pairing brittle; choosing just one slice may waste information, but using *all* slices is too heavy. The author therefore designs a **compromise multi-input CNN** that accepts more than one image while staying compute-feasible. [Medium](https://medium.com/swlh/osic-pulmonary-fibrosis-progression-5f06febebe0" \t "_blank)

**3) Model design: a three-branch network (two images + tabular)**

# Route — Multi-branch CNN + Tabular (predicts the slope “a”)

## What problem it solves

Builds a model that learns a patient-specific **rate of FVC change** (the slope **a** in the linear relation **FVC = a·Week + intercept**). This reframes the task from predicting raw FVC directly to predicting the **progression rate**, then using it to derive FVC for future weeks.

## Inputs

* **Two CT images** (two image branches), each a DICOM slice from the patient’s baseline CT. The author uses two images instead of one to capture more image information while staying within compute limits.
* **Tabular metadata**: columns from train.csv such as Age, Gender, SmokingStatus (and related fields).  
  Because patients have **unequal numbers of scans** and **unequal week counts**, feeding multiple images directly helps the CNN see more anatomy without sampling 10–20 slices (too costly).

## Architecture (step by step)

1. **Two image branches** → each branch processes **one DICOM image** through CNN layers.
2. **One tabular branch** → ingests patient metadata from train.csv.
3. **Fusion** → features from both image branches + the tabular branch are **concatenated**.
4. **Head** → fused features pass through dense layers to output a **single scalar: the slope a**.  
   Rationale: Many patients have hundreds of slices; two image inputs (“something is better than nothing”) let the model learn from more than a single slice without blowing the memory budget.

## Training & evaluation

* **Data generator** feeds batches without loading all images into RAM.
* The competition’s evaluation uses a **modified Laplace log-likelihood**; the article links to the metric and explains σ handling. After predicting **a**, you can reconstruct per-week FVC and compute the competition score.

## Output

* **One value per example**: the **slope a** (rate of FVC change).
* Predicted **FVC for any week** can be obtained via the linear relation **FVC = a·Week + intercept**.

## Strengths & caveats

* **Strengths:**
  + Uses **two** images to improve image signal under tight compute.
  + Cleanly targets **progression** (slope), which is central to the task.
* **Caveats:**
  + Still limited vs. using many slices or a full 3D model; sampling only two images may miss some pathology.
  + Needs a sensible way to set/learn the **intercept** to recover absolute FVC.

**4) Target formulation: predict slope “a” instead of FVC directly**

* Instead of directly predicting next-week FVC (or both FVC and Percent), we predicts a **slope coefficient a** in a simple linear model:

A black text on a white background

AI-generated content may be incorrect.

The idea: pulmonary fibrosis shows a **progressive** decline, so fitting a patient-specific linear trajectory (slope + intercept) can be more stable given the short, uneven time series. The model thus focuses on learning **progression rate** (the slope) from images + metadata; with a estimated, FVC for any week can be computed via the line.

**5) Training objective & metric alignment**

* **Loss during training:** Use the Kaggle **modified Laplace log-likelihood** for evaluation. While exact training loss details aren’t spelled out line-by-line in the article, the overall system is built to produce predictions that can be evaluated by that metric (i.e., FVC forecasts with associated **confidence/σ**).

**3, Pulmonary Fibrosis Progression Prognosis Using Machine Learning**

<https://www.researchgate.net/publication/352735714_Pulmonary_Fibrosis_Progression_Prognosis_Using_Machine_Learning>

Our approach uses a combined forecast of four algorithms: DNN (Deep Neural Network), GBDT (Gradient Boosting Decision Tree) [4], NGBoost (Natural Gradient Boosting) [5] and ElasticNet [6].

## 1) Data & Features

**Source columns (tabular only):**  
Patient, Weeks, FVC, Percent, Age, Sex, SmokingStatus (all from train.csv).

Note: The final method does **not** use CT image features; the approach is purely tabular.

**Feature handling (typical steps captured in the paper):**

* Normalize/standardize continuous fields (e.g., Age, Percent, Weeks).
* Encode categorical fields (Sex, SmokingStatus) into numeric form.
* Ensure **patient-wise splitting** so the same patient never appears in both train and validation sets.

## 2) Model Set (trained in parallel, all tabular)

The pipeline trains **four** complementary tabular models. Each learns to predict **FVC**; some also learn **uncertainty** directly:

1. **Deep Neural Network (DNN) with ordered outputs**
   * **Input:** engineered tabular features.
   * **Hidden:** standard dense layers (e.g., 100-unit ReLU blocks).
   * **Head (ordered triad):** the network outputs **three values** interpreted as:
     + **lower**, **middle** (point FVC), **upper**  
       enforced to be **monotonic** (lower ≤ middle ≤ upper) via a base + non-negative-increments trick (preds = p1 + cumsum(ReLU(p2))).
   * **Uncertainty:** A black text on a white background

     AI-generated content may be incorrect.
2. **LightGBM (GBDT) with Gaussian NLL**
   * Learns **mean** (FVC) and **variance** through a Gaussian **negative log-likelihood** objective.
   * **Uncertainty:** σ is derived from the predicted variance (via a positivity reparameterization like softplus).
3. **NGBoost (Natural-Gradient Boosting)**
   * Directly learns a **probability distribution** over FVC (e.g., Normal).
   * **Uncertainty:** σ comes from the model’s predicted distribution (e.g., from a central interval width).
4. **ElasticNet (linear with L1+L2)**
   * Linear baseline that is stable and interpretable.
   * **Uncertainty:** not predicted (mean FVC only).
   * Role in the ensemble: **variance reduction** and regularization.

## 3) Training Objective & Metric Alignment

**Competition-style metric (used for monitoring / auxiliary loss):**  
For each sample:

* **Point prediction:** use the **middle** output (DNN) or mean (GBDT/NGBoost/ElasticNet).
* **Uncertainty:** a σ per model when available; later combined (see Ensembling).
* Apply stabilizers consistent with OSIC practice:

A black text on a white background

AI-generated content may be incorrect.

* Metric (lower is better):

**A math symbols with black text

AI-generated content may be incorrect.**

**DNN loss (hybrid):**

* Combines **quantile/pinball losses** (to train lower/median/upper sensibly) with the **metric-like term** (to align uncertainty with the leaderboard score).
* This makes the DNN’s interval outputs both **ordered** and **calibrated**.

**Tree/boosting losses:**

* **LightGBM:** Gaussian NLL (fits mean and variance).
* **NGBoost:** distributional loss via natural gradients.

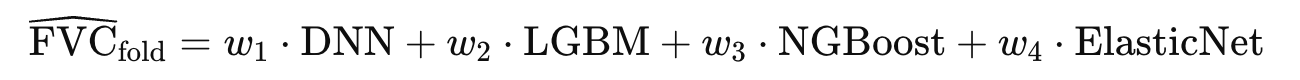
## 4) Cross-Validation Protocol (patient-wise)

* **5-fold CV** with **patient-level splits** (no leakage).
* For each fold:
  1. Train **all four models** on the training patients.
  2. Generate fold-out predictions (FVC and, where available, σ) for the validation/test patients.

## 5) Ensembling & Aggregation (single, coherent flow)

**Inside each fold:**

1. **Blend the FVC means (learned weights):**
   * On the fold’s validation set, solve a small **ordinary least squares** system to find **optimal linear weights** for the four model predictions:

This **data-driven** blend typically beats simple averaging.

1. **Aggregate uncertainty (σ):**
   * Take the **mean σ** across models that **predict σ** (DNN, LightGBM, NGBoost).
   * ElasticNet contributes only to the **mean FVC**, not σ.

**Across folds:**

* **Average** the fold-level blended **FVC** predictions to get the final mean FVC.
* **Average** the fold-level **σ** estimates to get the final σ.
* This two-level averaging (within fold → across folds) stabilizes both the central prediction and the uncertainty.

## 6) Inference & Submission

For every (Patient, Week) row required by the competition submission:

1. Build the **tabular feature vector** with the same preprocessing as training.
2. Run **all four trained models** to obtain:
   * FVC means from each model (and σ where available).
3. Apply the **learned fold-blend weights** to get the **final FVC**.
4. **Average σ** across probabilistic models; apply OSIC **σ clipping** for the metric/submission convention.
5. Write submission.csv with **FVC** and **Confidence** (derived from σ as per the competition format).

**4, Fibrosis-Net: A Tailored Deep Convolutional Neural Network Design for Prediction of Pulmonary Fibrosis Progression From Chest CT Images**

<https://pmc.ncbi.nlm.nih.gov/articles/PMC8596329/>

**Goal**  
Predict a patient’s **future FVC** (mL) at a requested timepoint, while staying **efficient** and **clinically grounded**. The model uses **CT slices** (visual evidence) and **clinical metadata** (age, sex, smoking) plus the **initial spirometry FVC**, and validates attention with **explainability**. (Intro & Method overview)

## 1) Inputs & Preprocessing

### 1.1 Data coming in

* **CT scan**: full axial stack per patient.
* **Clinical metadata**: age, sex, smoking status.
* **Initial spirometry**: baseline FVC measurement. (p5)

### 1.2 Slice selection (where PF lives)

* Keep only the **lower 55%** of slices (PF signs are more prevalent in the lower lungs). (p5)

Why this matters: lowers compute, and increases signal-to-noise by focusing on the anatomic region where fibrosis typically appears.

## 2) Model Architecture (Fibrosis-Net)

Think of Fibrosis-Net as **per-slice encoding → per-slice progression signal → patient-level fusion → final FVC combiner**. It’s a **tailored, lightweight CNN** with **late fusion** of clinical info and a small linear combiner at the end. (p5–p6)

### 2.1 Per-slice visual encoder (CNN)

For each slice in the **lower 55%**:

1. **Convolutional stack** encodes the slice into a **condensed feature vector**. (p5)
   * The backbone uses **lightweight components** (depthwise + pointwise convolutions and related efficient blocks) and **selective long-range connections** (sparse “connectivity hubs”) to be **compact but expressive**. (p5–p6)
2. You now have one **visual feature** per slice.

### 2.2 Slice-level “progression” head (rate regressor)

* **Concatenate** the slice’s visual feature with the **clinical metadata**.
* Feed into a **dense layer** that predicts a **linear rate of change in lung function** for that slice (intuitively, a “slope” signal). (p5)

Why “rate”? It aligns the representation with **progression** rather than just snapshot severity and makes patient-level fusion flexible. (p5)

### 2.3 Patient-level FVC prediction layer (late fusion)

Aggregate all slice-level rates for a patient and fuse with the clinical context:

1. **Median aggregation of slice rates**
   * Across the selected slices, take the **median** of the predicted slice rates. (p5)
2. **Fuse with initial spirometry + metadata**
   * Combine the **median rate**, the patient’s **initial FVC**, and **metadata (age, sex, smoking)** to form the input to the FVC prediction layer. (p5)
3. **Elastic Net combiner (clinical baseline)**
   * In parallel, fit a small **Elastic Net** regressor on **clinical metadata** to get a **second FVC estimate** (a stable clinical baseline). (p5)
4. **Final combination**
   * **Blend** the CNN-based FVC estimate with the **Elastic Net** estimate to produce the **final FVC** at the requested timepoint. (p5)

This “two-path combiner” improves **robustness**: the CNN brings image-derived progression cues; Elastic Net anchors the prediction in structured clinical priors. (p5)

## 3) Training & Inference Settings

* **Framework:** TensorFlow.
* **Optimizer:** **Adam**.
* **Loss:** **MAE** (mean absolute error).
* **Hyperparameters:** learning rate **1e-4**, **exponential decay 0.99 every 100 steps**, **batch size 8**. (p5–p6)
* **Hardware noted:** training on i9-9820X/RTX 2080 Ti; inference tested on CPU. (p5–p6)

Rationale: MAE is a simple, stable regression loss; the LR schedule + small batch suit the compact, efficient backbone.

## 4) Explainability-Driven Validation

* The authors audit predictions with **GSInquire** to verify that important PF-related regions (e.g., honeycombing-like patterns, subpleural involvement, etc.) are actually driving the model’s decisions. (pp6–9)
* This supports **“right-reasons”** behavior and increases clinical trust.

## 5) Efficiency & Design Choices (why this architecture)

* **Lower-55% slices:** focuses computation where fibrosis signal is strongest. (p5)
* **Lightweight blocks + sparse long-range connections:** keep parameters and latency low while preserving representational capacity. (p5–p6)
* **Late fusion with clinical info:** lets image features and clinical priors interact **after** the visual abstraction is learned. (p6)
* **Median rate aggregation:** robust to outlier slices; summarizes per-slice variability into a stable patient-level trend. (p5)
* **Elastic Net blend:** linear clinical baseline stabilizes final FVC, improving **generalization**. (p5)

## 6) Inference Flow (step-by-step)

1. **Load patient**: CT stack + metadata + initial FVC.
2. **Select slices**: keep **lower 55%** of the stack. (p5)
3. **Per slice**:
   * CNN → **visual feature**.
   * Concatenate with metadata → **dense** → **slice rate**. (p5)
4. **Aggregate**: median of slice rates → **patient-level rate**. (p5)
5. **FVC prediction layer**:
   * Combine **median rate + initial FVC + metadata** → **CNN FVC estimate**. (p5)
   * Elastic Net on metadata → **clinical FVC estimate**. (p5)
   * **Combine** both → **final FVC** at the requested timepoint. (p5)