Annotating biomedical images is very challenging.
It is not only tedious and time consuming, but also demanding of costly, specialty-oriented knowledge and skills, which are not easily accessible.

“How to Cut Annotation Cost?”
Key Ideas:
1. Active Selection
2. Handling noisy labels
3. Continuous fine-tuning

Three Applications:
1. Quality Assessment
2. Polyp Detection
3. Pulmonary Embolism Detection

Conclusion:
Our method can cut the annotation cost by >95% relative to training from scratch; >93% relative to full fine-tuning; and >60% relative to random selection.

Observation: To boost the performance of CNNs in biomedical imaging, multiple patches are usually generated automatically for each candidate through data augmentation; these patches generated from the same candidate share the same label, and are naturally expected to have similar predictions by the current CNN before they are expanded into the training dataset.
Active Selecting

Continuous Fine-tuning CNN₀

Active Selecting

Continuous Fine-tuning CNN₁

Testing CNN₂

...
Handling noisy labels via majority selection.
**Algorithm 1: Active incremental fine-tuning method.**

**Input:**
- $U = \{C_i\}, i \in [1, n] \{U$ contains $n$ candidates$\}$
- $C_i = \{x^j_i\}, j \in [1, m] \{C_i$ has $m$ patches$\}$
- $M_0$: pre-trained CNN
- $b$: batch size
- $\alpha$: patch selection ratio

**Output:**
- $L$: labeled candidates
- $M_t$: fine-tuned CNN model at Iteration $t$

**Functions:**
- $p \leftarrow P(C, M)$ \{outputs of $M$ given $\forall x \in C$\}
- $M_t \leftarrow F(L, M_{t-1})$ \{fine-tune $M_{t-1}$ with $L$\}
- $a \leftarrow mean(p_i)$ \{$a = \frac{1}{m} \sum_{j=1}^{m} p^j_i$\}

**Initialize:**
- $L \leftarrow \emptyset$, $t \leftarrow 1$

1. repeat
   2. for each $C_i \in U$ do
      3. $p_i \leftarrow P(C_i, M_{t-1})$
      4. if $mean(p_i) > 0.5$ then
         5. $S'_i \leftarrow$ top $\alpha$ percent of the patches of $C_i$
      6. else
         7. $S'_i \leftarrow$ bottom $\alpha$ percent of the patches of $C_i$
      end
   8. Build matrix $R_i$ using Eq. 3 for $S'_i$
   9. end
10. Sort $U$ according to the numerical sum of $R_i$
11. Query labels for top $b$ candidates, yielding $Q$
12. $L \leftarrow L \cup Q$; $U \leftarrow U \setminus Q$
13. $M_t \leftarrow F(L, M_{t-1})$; $t \leftarrow t + 1$
14. until *classification performance is satisfactory*;
Algorithm 1: Active incremental fine-tuning method.

Input:
\( \mathcal{U} = \{ \mathcal{C}_i \}, \ i \in [1, n] \) \{ \( \mathcal{U} \) contains \( n \) candidates \}
\( \mathcal{C}_i = \{ x^j_i \}, \ j \in [1, m] \) \{ \( \mathcal{C}_i \) has \( m \) patches \}
\( \mathcal{M}_0 \): pre-trained CNN
\( b \): batch size
\( \alpha \): patch selection ratio

Output:
\( \mathcal{L} \): labeled candidates
\( \mathcal{M}_t \): fine-tuned CNN model at Iteration \( t \)

Functions:
\( p \leftarrow P(\mathcal{C}, \mathcal{M}) \) \{ outputs of \( \mathcal{M} \) given \( \forall x \in \mathcal{C} \) \}
\( \mathcal{M}_t \leftarrow F(\mathcal{L}, \mathcal{M}_{t-1}) \) \{ fine-tune \( \mathcal{M}_{t-1} \) with \( \mathcal{L} \) \}
\( a \leftarrow \text{mean}(p_i) \) \{ \( a = \frac{1}{m} \sum_{j=1}^{m} p^j_i \) \}

Initialize:
\( \mathcal{L} \leftarrow \emptyset \), \( t \leftarrow 1 \)

1 repeat
   
   for each \( \mathcal{C}_i \in \mathcal{U} \) do
      
      \( p_i \leftarrow P(\mathcal{C}_i, \mathcal{M}_{t-1}) \)
      
      if \( \text{mean}(p_i) > 0.5 \) then
         
         \( S_i' \leftarrow \text{top} \ \alpha \ \text{percent of the patches of} \ \mathcal{C}_i \)
      
      else
         
         \( S_i' \leftarrow \text{bottom} \ \alpha \ \text{percent of the patches of} \ \mathcal{C}_i \)
      
   end

   Build matrix \( R_i \) using Eq. 3 for \( S_i' \)

10 end

11 Sort \( \mathcal{U} \) according to the numerical sum of \( R_i \)
12 Query labels for top \( b \) candidates, yielding \( \mathcal{Q} \)
13 \( \mathcal{L} \leftarrow \mathcal{L} \cup \mathcal{Q} \); \( \mathcal{U} \leftarrow \mathcal{U} \setminus \mathcal{Q} \)
14 \( \mathcal{M}_t \leftarrow F(\mathcal{L}, \mathcal{M}_{t-1}) \); \( t \leftarrow t + 1 \)
15 until classification performance is satisfactory;
Algorithm 1: Active incremental fine-tuning method.

Input:
\( \mathcal{U} = \{C_i\}, i \in [1, n] \) \( \{ \mathcal{U} \) contains \( n \) candidates \)
\( C_i = \{x_{ij}^i\}, j \in [1, m] \) \( \{ C_i \) has \( m \) patches \)
\( \mathcal{M}_0 \): pre-trained CNN
\( b \): batch size
\( \alpha \): patch selection ratio

Output:
\( \mathcal{L} \): labeled candidates
\( \mathcal{M}_t \): fine-tuned CNN model at Iteration \( t \)

Functions:
\( p \leftarrow P(C, \mathcal{M}) \) \( \{ \text{outputs of } \mathcal{M} \text{ given } \forall x \in C \} \)
\( \mathcal{M}_t \leftarrow F(\mathcal{L}, \mathcal{M}_{t-1}) \) \( \{ \text{fine-tune } \mathcal{M}_{t-1} \text{ with } \mathcal{L} \} \)
\( a \leftarrow \text{mean}(p_i) \) \( \{ a = \frac{1}{m} \sum_{j=1}^{m} p_{ij}^i \} \)

Initialize:
\( \mathcal{L} \leftarrow \emptyset \), \( t \leftarrow 1 \)

1. repeat
2. for each \( C_i \in \mathcal{U} \) do
3. \( p_i \leftarrow P(C_i, \mathcal{M}_{t-1}) \)
4. if \( \text{mean}(p_i) > 0.5 \) then
5. \( S_i' \leftarrow \text{top } \alpha \text{ percent of the patches of } C_i \)
6. else
7. \( S_i' \leftarrow \text{bottom } \alpha \text{ percent of the patches of } C_i \)
8. end
9. Build matrix \( R_i \) using Eq. 3 for \( S_i' \)
10. end
11. Sort \( \mathcal{U} \) according to the numerical sum of \( R_i \)
12. Query labels for top \( b \) candidates, yielding \( \mathcal{Q} \)
13. \( \mathcal{L} \leftarrow \mathcal{L} \cup \mathcal{Q} \); \( \mathcal{U} \leftarrow \mathcal{U} \setminus \mathcal{Q} \)
14. \( \mathcal{M}_t \leftarrow F(\mathcal{L}, \mathcal{M}_{t-1}) \); \( t \leftarrow t + 1 \)
15. until classification performance is satisfactory;
**Algorithm 1: Active incremental fine-tuning method.**

**Input:**
- $\mathbf{U} = \{C_i\}$, $i \in [1, n]$ {\(\mathbf{U}\) contains \(n\) candidates}
- $C_i = \{x^j_i\}$, $j \in [1, m]$ {\(C_i\) has \(m\) patches}
- $\mathcal{M}_0$: pre-trained CNN
- $b$: batch size
- $\alpha$: patch selection ratio

**Output:**
- $\mathcal{L}$: labeled candidates
- $\mathcal{M}_t$: fine-tuned CNN model at Iteration \(t\)

**Functions:**
- $p \leftarrow P(\mathcal{C}, \mathcal{M})$ {outputs of $\mathcal{M}$ given $\forall x \in \mathcal{C}$}
- $\mathcal{M}_t \leftarrow F(\mathcal{L}, \mathcal{M}_{t-1})$ {fine-tune $\mathcal{M}_{t-1}$ with $\mathcal{L}$}
- $a \leftarrow \text{mean}(p_t)$ {\(a = \frac{1}{m} \sum_{j=1}^{m} p^j_t\)}

**Initialize:**
- $\mathcal{L} \leftarrow \emptyset$, $t \leftarrow 1$

1. \(\text{repeat}\)
   2. \(\text{repeat}\)
      3. \(p_t \leftarrow P(C_i, \mathcal{M}_{t-1})\)
      4. \(\text{if} \ \text{mean}(p_t) > 0.5 \ \text{then}\)
         5. \(S'_t \leftarrow \text{top } \alpha \% \text{ of the patches of } C_i\)
      6. \(\text{else}\)
         7. \(S'_t \leftarrow \text{bottom } \alpha \% \text{ of the patches of } C_i\)
      8. \(\text{end}\)
   9. \(\text{Build matrix } R_t \text{ using Eq. 3 for } S'_t\)
  10. \(\text{end}\)
  11. Sort $\mathbf{U}$ according to the numerical sum of $R_t$
  12. Query labels for top $b$ candidates, yielding $\mathcal{Q}$
  13. $\mathcal{L} \leftarrow \mathcal{L} \cup \mathcal{Q}$; $\mathbf{U} \leftarrow \mathbf{U} \setminus \mathcal{Q}$
  14. $\mathcal{M}_t \leftarrow F(\mathcal{L}, \mathcal{M}_{t-1})$; $t \leftarrow t + 1$
  15. \(\text{until classification performance is satisfactory};\)
**Algorithm 1: Active incremental fine-tuning method.**

**Input:**
- \( U = \{C_i\}, \ i \in [1, n] \) \{\( U \) contains \( n \) candidates\}
- \( C_i = \{x^j_i\}, \ j \in [1, m] \) \{\( C_i \) has \( m \) patches\}
- \( M_0 \): pre-trained CNN
- \( b \): batch size
- \( \alpha \): patch selection ratio

**Output:**
- \( \mathcal{L} \): labeled candidates
- \( \mathcal{M}_t \): fine-tuned CNN model at Iteration \( t \)

**Functions:**
- \( p \leftarrow P(C, M) \) \{outputs of \( M \) given \( \forall x \in C \)\}
- \( M_t \leftarrow F(\mathcal{L}, M_{t-1}) \) \{fine-tune \( M_{t-1} \) with \( \mathcal{L} \)\}
- \( a \leftarrow \text{mean}(p_i) \) \( a = \frac{1}{m} \sum_{j=1}^{m} p^j_i \)

**Initialize:**
- \( \mathcal{L} \leftarrow \emptyset \), \( t \leftarrow 1 \)

1. **repeat**
2. \hspace{1em} **for each** \( C_i \in U \) **do**
3. \hspace{2em} \( p_i \leftarrow P(C_i, M_{t-1}) \)
4. \hspace{3em} **if** \( \text{mean}(p_i) > 0.5 \) **then**
5. \hspace{4em} \( S'_i \leftarrow \text{top } \alpha \text{ percent of the patches of } C_i \)
6. \hspace{3em} **else**
7. \hspace{4em} \( S'_i \leftarrow \text{bottom } \alpha \text{ percent of the patches of } C_i \)
8. \hspace{3em} **end**
9. \hspace{2em} Build matrix \( R_i \) using Eq. 3 for \( S'_i \)
10. **end**
11. Sort \( U \) according to the numerical sum of \( R_i \)
12. **Query** labels for top \( b \) candidates, yielding \( \mathcal{Q} \)
13. \( \mathcal{L} \leftarrow \mathcal{L} \cup \mathcal{Q}; \ U \leftarrow U \setminus \mathcal{Q} \)
14. \( \mathcal{M}_t \leftarrow F(\mathcal{L}, \mathcal{M}_{t-1}); t \leftarrow t + 1 \)
15. **until** classification performance is satisfactory;
Quality Assessment

Polyp Detection

Pulmonary Embolism Detection