

Deep Dive: Core Concepts

Understanding the Why Behind Each Design Decision

Deep Dive 1: Self-Supervised Learning with Progress & Elapsed Time

Elapsed Time vs Progress: The Critical Difference

These are TWO DIFFERENT concepts serving DIFFERENT purposes:

ELAPSED TIME:

- What it is: How long since surgery started (e.g., 20 minutes)
- How we get it: From video timestamp
- Used as: INPUT to the model (helps it know "when" things happen)

PROGRESS:

- What it is: Fraction of surgery complete (0.0 to 1.0)
- How we get it: Calculated as elapsed/total duration
- Used as: TARGET to predict (self-supervised learning signal)

Why Elapsed Time as Input is Crucial

THE PROBLEM: A frame showing "surgeon using grasper" is ambiguous.

- Could be minute 5 (early) --> 35 min remaining
- Could be minute 45 (late) --> 5 min remaining
- Same visual, completely different predictions needed!

THE SOLUTION: Tell the model what time it is!

By concatenating elapsed time with visual features, the model learns time-conditioned patterns:

- "Grasper + elapsed=0.1 (6 min) --> remaining is approximately 35 min"
- "Grasper + elapsed=0.8 (48 min) --> remaining is approximately 3 min"

The visual appearance PLUS temporal context enables accurate prediction.

Why Progress Prediction Helps (Self-Supervised Learning)

What is self-supervised learning?

- Regular supervised: Human annotates labels (expensive, slow, limited)
- Self-supervised: Labels come FREE from the data itself (scalable!)

Progress = $\text{elapsed_time} / \text{total_duration}$

This label costs NOTHING - we compute it automatically from timestamps!

By training to predict progress, the model MUST learn:

- Which phase we are in (early phases = low progress)
- How phases typically progress over time
- Visual cues that indicate progression
- What a "normal" surgery timeline looks like

This understanding TRANSFERS directly to time prediction!

Surgical Duration Prediction - Deep Learning Guide

The Mathematical Connection

```
progress = elapsed / total
```

Rearranging algebraically:

```
total = elapsed / progress
```

```
remaining = total - elapsed
```

```
remaining = elapsed * (1 - progress) / progress
```

EXAMPLE:

- Elapsed = 20 minutes

- Model predicts progress = 0.60 (60% complete)

- Remaining = $20 * (1 - 0.60) / 0.60 = 13.3$ minutes

If the model learns progress well, it has a strong prior for remaining time!

Deep Dive 2: How Different Losses Combine in Backpropagation

The Four Losses Explained

1. CrossEntropyLoss (for phase classification)
 - Measures how wrong our phase prediction probability is
 - Confident and correct (P=0.99): loss = 0.01 (very small)
 - Uncertain (P=0.50): loss = 0.69 (medium)
 - Wrong (P=0.01): loss = 4.6 (large penalty!)
2. L1Loss (for time and progress predictions)
 - Simply the absolute difference in predicted vs actual
 - Predicted=12.3 min, True=15.0 min: loss = 2.7
 - Robust to outliers (unlike L2/MSE)
 - Clinically interpretable: "We are off by X minutes on average"

Why Weighted Combination?

```
total_loss = 0.3 * phase_loss      # Important for understanding workflow
              + 0.2 * phase_time_loss # Secondary objective
              + 0.3 * surgery_time_loss # Main objective (highest priority)
              + 0.2 * progress_loss   # Self-supervised signal
```

Weights serve two purposes:

1. SCALE BALANCING: Different losses have different magnitudes
 - Phase CE: typically 0.5 - 2.0
 - Surgery time L1: typically 5.0 - 20.0
 - Without weights, larger losses would dominate
2. IMPORTANCE: We want to prioritize certain objectives
 - Surgery time gets 0.3 (main goal)
 - Phase classification gets 0.3 (helps everything)
 - Others get 0.2 (supporting roles)

How Gradients Flow Through the Network

During backpropagation, gradients from ALL losses flow through the ENTIRE shared network:

The gradient of total_loss with respect to any weight equals:

$0.3 * (\text{gradient from phase_loss}) +$
 $0.2 * (\text{gradient from phase_time_loss}) +$
 $0.3 * (\text{gradient from surgery_time_loss}) +$
 $0.2 * (\text{gradient from progress_loss})$

This means:

CNN receives gradients saying:

- "Extract features that help classify phase"
- "Extract features that help predict time"
- "Extract features that help predict progress"

LSTM receives gradients saying:

- "Learn temporal patterns for phase transitions"
- "Learn temporal patterns for time estimation"
- "Learn temporal patterns for progress tracking"

Surgical Duration Prediction - Deep Learning Guide

The network learns features useful for ALL tasks simultaneously!

Deep Dive 3: How CNN Features + Elapsed Time Flow Through LSTM

Step 1: CNN Extracts Visual Features

Input: Video frame [3, 224, 224] - raw RGB pixels

Output: Feature vector [2048] - semantic description

The CNN (ResNet-50) transforms raw pixels into meaningful numbers that encode:

- Tool presence and position
- Tissue appearance and color
- Anatomical structures visible
- Surgical actions being performed

These 2048 numbers are a compressed "understanding" of the image.

Step 2: Concatenate Elapsed Time

```
visual_features = CNN(frame)          # [2048] - WHAT we see
elapsed_normalized = [0.52]           # [1] - WHEN we see it (52% of expected duration)
combined = concat(visual, elapsed)    # [2049] - Complete context
```

This simple concatenation allows the network to learn ANY function that combines visual and temporal information.

Step 3: LSTM Processes the Sequence

The LSTM receives 30 frames, each with 2049 features.

At each timestep, the LSTM cell:

1. Receives: current input (frame features) + previous memory
2. Decides through learned gates:
 - What to FORGET from old memory
 - What to ADD from new input
 - What to OUTPUT
3. Produces: updated memory + hidden state output

After processing all 30 frames, the final hidden state [256] encodes:

- What tools/anatomy were visible across the 30-second window
- How the scene changed over time
- The elapsed time context at each moment
- Learned patterns that correlate with remaining time

Why This Architecture Enables Accurate Prediction

WITHOUT elapsed time:

- Model sees identical grasper images from two surgeries
- Surgery A: grasper at minute 10, 35 min remaining
- Surgery B: grasper at minute 50, 5 min remaining
- Same input, different outputs needed --> IMPOSSIBLE

WITH elapsed time:

- Model receives: grasper features + elapsed=0.17 --> learns "early, lots remaining"
- Model receives: grasper features + elapsed=0.83 --> learns "late, almost done"
- Different inputs for different situations --> LEARNABLE

The key insight: The LSTM learns to combine:

1. WHAT it sees (from CNN visual features)

Surgical Duration Prediction - Deep Learning Guide

2. WHEN it sees it (from elapsed time)
3. HOW things changed (from processing the sequence)

This three-way combination enables accurate, context-aware time prediction!