CTC Loss

Connectionist Temporal Classification

Alignment-Free Sequence Labeling

No need for frame-level annotations

🎯 Alignment-Free

Handles sequences without explicit alignment

📝 No Frame Annotations

Only sequence-level labels needed

⚪ Blank Token

Introduces blank for alignment flexibility

🔄 Variable Length

Input/output can have different lengths

🔍 How It Works

📊 Probability Distribution Generation

The neural network outputs a probability distribution over all possible characters (alphabet) at each time step, including a special 'blank' token.

🔀 Alignment Paths

Multiple alignment paths can produce the same output sequence. For example, "CAT" can be represented as "C-A-T", "CC-AAT", "-C-A-T-", and many other variations.

💡 Key Idea

CTC sums the probabilities of all possible alignment paths that can be converted to the target sequence.

⚙️ Computation Methodology

Step 1: Forward Algorithm

Progressing from left to right, at each time step, cumulatively calculate the probabilities of all paths that can reach each position in the target sequence.

Step 2: Backward Algorithm

Progressing from right to left, calculate the probabilities of all paths that reach the end of the sequence from each position.

Step 3: Marginalization

Combine forward and backward probabilities to calculate the total probability of all possible alignments that generate the target sequence. This is efficiently performed using dynamic programming.

📉 Loss Function Calculation

CTC Loss = -log(Sum of probabilities of all correct alignment paths)

The training objective is to minimize this loss, thereby maximizing the probability of the correct output sequence.

📘 Practical Calculation Example

🔢 Problem Setup

Target Output: "A" (1D sequence)

Input: 3 time steps (T=3)

Possible Characters: {A, blank(-)}

📊 Neural Network Output (Probability Distribution)

Time Step	P(A)	P(blank)
t=1	0.4	0.6
t=2	0.7	0.3
t=3	0.5	0.5

🌲 All Possible Alignment Paths

Calculate all possible paths and their probabilities that can produce the output "A":

Path 1: A - -

→ P(A at t=1) × P(- at t=2) × P(- at t=3)

→ 0.4 × 0.3 × 0.5 = **0.060**

Path 2: - A -

→ P(- at t=1) × P(A at t=2) × P(- at t=3)

→ 0.6 × 0.7 × 0.5 = **0.210**

Path 3: - - A

→ P(- at t=1) × P(- at t=2) × P(A at t=3)

→ 0.6 × 0.3 × 0.5 = **0.090**

Path 4: A A -

→ P(A at t=1) × P(A at t=2) × P(- at t=3)

→ 0.4 × 0.7 × 0.5 = **0.140**

Path 5: A - A

→ P(A at t=1) × P(- at t=2) × P(A at t=3)

→ 0.4 × 0.3 × 0.5 = **0.060**

Path 6: - A A

→ P(- at t=1) × P(A at t=2) × P(A at t=3)

→ 0.6 × 0.7 × 0.5 = **0.210**

Path 7: A A A

→ P(A at t=1) × P(A at t=2) × P(A at t=3)

→ 0.4 × 0.7 × 0.5 = **0.140**

✅ Final CTC Loss Calculation

Step 1: Sum probabilities of all paths

P(A) = 0.060 + 0.210 + 0.090 + 0.140 + 0.060 + 0.210 + 0.140 = **0.910**

Step 2: Calculate negative log probability

CTC Loss = -log(0.910) = **0.094**

💡 Key Insights

• High probability paths (0.210, 0.140) significantly influence the overall loss
• By considering all possible alignments, explicit alignment information is not required
• Through training, the total probability of correct outputs approaches 1 (Loss → 0)

🎯 Decoding Strategies

🏃 Greedy Decoding

Select the character with the highest probability at each time step, then remove consecutive duplicate characters and blanks. Fast but not always optimal.

🔍 Beam Search

Explore multiple candidate paths simultaneously to find the sequence with the highest overall probability. More accurate but computationally expensive.

🎓 Training Approach

Marginalizes over all possible alignments using dynamic programming

💼 Primary Applications

Speech Recognition Handwriting Recognition