Complete Taxonomy of Fixes for Caption Generation Mode Collapse

Overview

Mode collapse = Model generates same/similar captions for all inputs. Here are ALL the fixes, organized by what they target.

Category 1: DECODING STRATEGIES (Inference-Time Fixes)

Changes how you generate text from the trained model - NO retraining needed

1.1 Temperature Sampling

What it does: Makes the model less confident, adds randomness

```python
# Instead of: next_word = logits.argmax()
logits = logits / temperature  # temperature > 1 = more random
probs = F.softmax(logits, dim=-1)
next_word = torch.multinomial(probs, 1)
```

Pros:

- ✅ Instant fix, no retraining
- ✅ Easy to implement
- ✅ Can tune temperature per sample

Cons:

- ❌ Can generate nonsense if temperature too high
- ❌ Doesn't fix underlying model problems

Best for: Quick test to see if sampling helps

Temperature guide:

- `0.1-0.5`: Very conservative, focused
- `0.7-1.0`: **Recommended range** for captions
- `1.0-1.5`: More creative, riskier
- `>2.0`: Usually too random

1.2 Top-k Sampling

What it does: Only consider top k most likely words

```python
top_k = 50  # Only look at 50 most likely words
values, indices = torch.topk(logits, top_k)
logits_filtered = torch.full_like(logits, float('-inf'))
logits_filtered.scatter_(1, indices, values)
probs = F.softmax(logits_filtered, dim=-1)
next_word = torch.multinomial(probs, 1)
```

Pros:

- ✅ Prevents extremely unlikely words
- ✅ More controlled than pure temperature

Cons:

- ❌ Fixed k doesn't adapt to context
- ❌ Can still generate repetitive if k too small

Best for: Combined with temperature

k values:

- `k=10-20`: Very safe
- `k=40-60`: **Recommended**
- `k=100+`: Similar to no filtering

1.3 Top-p (Nucleus) Sampling

What it does: Consider smallest set of words whose cumulative probability > p

```python
top_p = 0.9  # Consider top 90% probability mass
sorted_logits, sorted_indices = torch.sort(logits, descending=True)
cumulative_probs = torch.cumsum(F.softmax(sorted_logits, dim=-1), dim=-1)

# Remove tokens after threshold
remove_mask = cumulative_probs > top_p
remove_mask[..., 1:] = remove_mask[..., :-1].clone()
remove_mask[..., 0] = 0

logits[sorted_indices[remove_mask]] = float('-inf')
probs = F.softmax(logits, dim=-1)
next_word = torch.multinomial(probs, 1)
```

Pros:

- ✅ Adapts to context (sometimes considers 5 words, sometimes 50)
- ✅ Better than top-k for most tasks
- ✅ Industry standard

Cons:

- ❌ More complex to implement
- ❌ Slightly slower

Best for: Production caption generation

p values:

- `0.8-0.85`: Conservative
- `0.9-0.95`: **Recommended**
- `0.95-0.99`: More diverse

1.4 Beam Search

What it does: Keep multiple candidate sequences, pick best overall

```python
beam_size = 5
# Track top 5 sequences at each step
# Score = sum of log probabilities
# Return highest scoring complete sequence
```

Pros:

- ✅ More coherent than sampling
- ✅ Finds globally better sequences
- ✅ Deterministic (reproducible)

Cons:

- ❌ Can still be repetitive
- ❌ Much slower (5x for beam_size=5)
- ❌ Complex to implement

Best for: When you need one "best" caption

Beam sizes:

- `3-5`: **Standard**
- `10+`: Diminishing returns, very slow

1.5 Diverse Beam Search

What it does: Beam search but penalizes similar beams

```python
# Each beam has diversity penalty
# Beams that generate similar tokens get lower scores
```

Pros:

- ✅ Combines beam search quality with diversity
- ✅ Can generate multiple diverse captions

Cons:

- ❌ Complex implementation
- ❌ Even slower than regular beam search
- ❌ Many hyperparameters to tune

Best for: When you need N diverse captions for one audio

Category 2: ARCHITECTURE FIXES (Model Design)

Changes to model structure - REQUIRES retraining

2.1 Better Audio Encoder

Problem: MaxPool destroys temporal information

```python
# BAD:
MaxPool2d(kernel_size=2, stride=2)  # Loses 50% of info each time

# GOOD:
Conv2d(stride=(2, 1))  # Pool frequency, keep time
AdaptiveAvgPool2d((1, None))  # Keep temporal dimension
```

Impact: ⭐⭐⭐⭐⭐ High **Effort:** Low (change a few lines)

2.2 Audio Injection Strategy

Problem: Audio only used once (initial hidden state)

```python
# BAD: Audio forgotten after first step
h0 = audio_features  # Only used here
gru_out, _ = gru(word_embeddings, h0)

# GOOD: Audio at every step
for t in range(seq_len):
    input_t = concat(word_embedding, audio_features)
    gru_out, h = gru(input_t, h)
```

Impact: ⭐⭐⭐⭐⭐ High **Effort:** Low-Medium

2.3 Attention Mechanism

What it does: Let decoder focus on different audio parts per word

```python
# At each decoding step:
# 1. Compute attention over audio time steps
# 2. Get weighted audio context
# 3. Use context to generate word
```

Impact: ⭐⭐⭐⭐ Medium-High **Effort:** Medium (need to implement attention)

When to use: If simpler fixes don't work

2.4 Deeper Encoder

What it does: More convolutional layers = better features

```python
# 2 layers → 3-4 layers
# Add residual connections
```

Impact: ⭐⭐⭐ Medium **Effort:** Low

Tradeoff: More parameters, slower training

2.5 Transformer Architecture

What it does: Replace CNN+RNN with full transformer

Impact: ⭐⭐⭐⭐⭐ High (if enough data) **Effort:** High (major rewrite)

Requirements:

- Large dataset (Clotho might be too small)
- More compute
- Careful hyperparameter tuning

Category 3: TRAINING TECHNIQUES

Changes to training process - REQUIRES retraining

3.1 Label Smoothing

What it does: Prevent overconfident predictions

```python
# Instead of: target = [0, 0, 1, 0, 0]  (one-hot)
# Use:        target = [0.02, 0.02, 0.92, 0.02, 0.02]

criterion = nn.CrossEntropyLoss(label_smoothing=0.1)
```

Impact: ⭐⭐⭐ Medium **Effort:** One line change

Values:

- `0.0`: No smoothing
- `0.05-0.15`: **Recommended**
- `0.2+`: Too much smoothing

3.2 Scheduled Sampling

What it does: During training, sometimes use predicted words instead of ground truth

```python
# Reduces exposure bias (train/test mismatch)
if random.random() < scheduled_sampling_rate:
    input_word = predicted_word  # What model generated
else:
    input_word = ground_truth_word  # From dataset
```

Impact: ⭐⭐⭐⭐ Medium-High **Effort:** Medium

Schedule:

- Start: 0% (use ground truth)
- End: 30-50% (use predictions)
- Gradually increase over epochs

3.3 Diversity Loss

What it does: Penalize generating same word repeatedly

```python
diversity_penalty = count_consecutive_repeats(predictions)
total_loss = cross_entropy_loss + alpha * diversity_penalty
```

Impact: ⭐⭐ Low-Medium **Effort:** Low-Medium

Cons: Can hurt caption quality if alpha too high

3.4 Data Augmentation

What it does: Increase training diversity

```
python

# Audio augmentation:
- Time stretching (0.8x - 1.2x speed)
- Pitch shifting (±2 semitones)
- Add background noise
- SpecAugment (mask frequency/time regions)

# Caption augmentation:
- Use all 5 captions per audio (not just one)
- Synonym replacement
- Paraphrase generation
```

Impact: ⭐⭐⭐⭐ Medium-High **Effort:** Medium-High

3.5 Curriculum Learning

What it does: Train on easy examples first, gradually harder

```
python

# Epoch 1-10: Short captions only
# Epoch 11-20: Medium captions
# Epoch 21+: All captions
```

Impact: ⭐⭐ Low-Medium **Effort:** Medium

Best for: Very large models or datasets

Category 4: DATA FIXES

Changes to dataset - one-time preprocessing

4.1 Class Balancing

Problem: "person walking" appears 1000x, "alarm clock" 10x

```python
# Oversample rare caption types
# Undersample common ones
# Or use weighted sampling
```

Impact: ⭐⭐⭐⭐ Medium-High **Effort:** Low-Medium

4.2 Caption Filtering

Problem: Some captions are too generic/vague

```python
# Remove captions like:
# "a sound is heard"
# "something is making noise"
# Keep specific ones
```

Impact: ⭐⭐⭐ Medium **Effort:** Low (manual work)

4.3 Use All 5 Captions

Problem: Using only first caption per audio

```python
# Your code does this - GOOD!
captions.extend(item["captions"])  # Use ALL
```

Impact: ⭐⭐⭐⭐ Medium-High **Effort:** Already done ✓

Category 5: REGULARIZATION

Prevent overfitting - REQUIRES retraining

5.1 Dropout

```python
nn.Dropout(p=0.3-0.5)  # In decoder
nn.Dropout2d(p=0.2-0.3)  # In CNN
```

Impact: ⭐⭐⭐ Medium **Effort:** One line

Your model: Already has this ✓

5.2 Weight Decay

```python
optimizer = AdamW(params, weight_decay=1e-4)
```

Impact: ⭐⭐ Low-Medium **Effort:** One parameter

Your model: Already has this ✓

5.3 Gradient Clipping

```python
torch.nn.utils.clip_grad_norm_(model.parameters(), max_norm=5.0)
```

Impact: ⭐⭐⭐ Medium (prevents training instability) **Effort:** One line

Your model: Already has this ✓

RECOMMENDED FIX STRATEGY

🔥 PRIORITY 1: Quick Wins (Do First)

1. **Temperature Sampling** (0.7-1.0) - 5 min, no retraining
2. **Top-p Sampling** (0.9) - 10 min, no retraining
3. **Test current model** with sampling → If works, great!

🎯 PRIORITY 2: Architecture Fixes (If Priority 1 fails)

4. **Fix Audio Encoder** - Remove aggressive MaxPool
5. **Audio at Every Step** - Concatenate with word embeddings
6. **Retrain** baseline with these fixes

🚀 PRIORITY 3: Advanced (If still not good)

7. **Attention Mechanism** - Add Bahdanau attention
8. **Scheduled Sampling** - Reduce exposure bias
9. **Data Augmentation** - SpecAugment

💪 PRIORITY 4: Heavy Lifting (Final resort)

10. **Transformer Architecture** - Full rewrite

11. **Pre-trained Models** - Use PANNs or AST

Quick Decision Tree

```
Is model generating same caption for everything?
│
├─ YES → Try temperature sampling (0.8) + top-p (0.9)
│        │
│        ├─ WORKS → Keep it! Try optimizing temperature
│        │
│        └─ STILL BROKEN → Retrain with:
│                          - Fixed audio encoder (no MaxPool on time)
│                          - Audio at every decoding step
│                          - Label smoothing (0.1)
│
└─ NO, but captions are generic/boring
         │
         └─ Try:
           - Lower temperature (0.7)
           - Beam search (size=5)
           - Attention mechanism
```

What to Do NOW

Step 1: Test sampling with your current trained model

```python
# Change one line in generate():
# next_word = logits.argmax(dim=1)  # OLD
probs = F.softmax(logits / 0.8, dim=-1)  # NEW
next_word = torch.multinomial(probs, 1).squeeze(1)
```

Step 2: If that doesn't work, retrain with architecture fixes

Step 3: Add attention if still not satisfied

Summary Table

Fix	Impact	Effort	Retrain?	Use When
Temperature sampling	⭐⭐⭐	5 min	No	First try
Top-p sampling	⭐⭐⭐⭐	10 min	No	First try
Beam search	⭐⭐⭐	30 min	No	Need best caption
Better encoder	⭐⭐⭐⭐⭐	10 min	Yes	Sampling fails
Audio at each step	⭐⭐⭐⭐⭐	20 min	Yes	Sampling fails
Attention	⭐⭐⭐⭐	2 hours	Yes	Above fails
Label smoothing	⭐⭐⭐	1 min	Yes	Easy addition
Scheduled sampling	⭐⭐⭐⭐	1 hour	Yes	Advanced
Transformer	⭐⭐⭐⭐⭐	1 day	Yes	Nothing else works