Model v6 Explained Simply

Image Captioning with Neural Networks

1 The Big Picture

Imagine you're teaching a friend to describe photos. You'd show them pictures and tell them what's in each one. Eventually, they'd learn patterns: "If there's grass and an animal, it's probably outdoors. If the animal has four legs and fur, say 'dog' or 'cat'."

Our model does exactly this, but with numbers instead of intuition.

1.1 The Challenge

- **Input:** A photo (millions of pixels: tiny colored dots)
- **Output:** A sentence like "a dog running through grass"
- **Problem:** How do we go from dots to words?

Key Idea: We can't just memorize "this exact photo = this exact sentence" because we'll never see the same photo twice during testing. We need to learn *patterns* that work for any photo.

2 The Two-Part Solution

Think of our model like a two-person team:

2.1 Person 1: The Observer (Encoder)

Job: Look at the photo and write down what you see

- Takes the photo
- Recognizes objects: "I see a dog, grass, sky"
- Notes details: "The dog is brown, the grass is green"
- Writes it all down as a list of 256 numbers

Example: If the photo has a dog, number 42 might be high. If there's grass, number 157 might be high. These numbers are like a secret code summarizing the image.

2.2 Person 2: The Writer (Decoder)

Job: Read the Observer's notes and write a sentence

- Reads the 256 numbers
- Generates words one at a time
- Checks: "Does this word make sense after the previous words?"

- Stops when the sentence is complete

Example: The Writer reads the notes: "High dog score, high grass score, high motion score" and generates: "a" → "dog" → "runs" → "on" → "grass"

3 How the Observer (Encoder) Works

3.1 Starting Point: Pre-trained Vision

We use something called **EfficientNet-B0**. Think of it as a trained expert who already knows what objects look like.

Key Idea: Instead of teaching our model "what is a dog?" from scratch, we hire an expert who already knows. This expert was trained on 1.2 million images, so it already understands dogs, cats, cars, trees, etc.

3.2 The Process

1. **Resize photo:** Make it 224×224 pixels (like standardizing paper size)
2. **Extract features:** The expert looks at it and produces 1,280 numbers

   Note: These 1,280 numbers capture everything: colors, shapes, objects, textures. But 1,280 is too many for our Writer to handle efficiently.
3. **Compress to 256 numbers:** Keep only the most important information

   Example: Like taking notes from a long book. Instead of writing down every sentence, you write the key points.
4. **Add noise (Dropout):** During training, randomly erase 30% of numbers

   Key Idea: This forces the model to not rely too heavily on any single number. If number 42 (dog detector) gets erased, the model learns to also use number 89 (animal detector) as backup. This prevents memorization.

3.3 Why 256 Numbers?

- **Too few (e.g., 64):** Can't capture enough detail. "Is it a dog or a cat?" becomes unclear.
- **Too many (e.g., 512):** The model memorizes specific images instead of learning patterns.
- **256 (our choice):** Just right for 8,000 training images.

Note: This is like choosing how detailed your notes should be. Too brief and you lose important info. Too detailed and you're just copying the book.

4 How the Writer (Decoder) Works

The Writer generates the caption word by word, like filling in a crossword puzzle one letter at a time.

4.1 The Generation Process

Step-by-step example:

1. **Start:** Begin with a special START signal
2. **Read notes:** Look at the 256 numbers from the Observer
3. **First word:** "Which word usually starts sentences?"
   - Model sees: high dog score, outdoor setting
   - Predicts: "a" (75% confident)
4. **Second word:** "What comes after 'a'?"
   - Looks at: image notes + previously written "a"
   - Predicts: "dog" (82% confident)
5. **Third word:** "What comes after 'a dog'?"
   - Looks at: image notes + "a dog"
   - Sees high motion score in image
   - Predicts: "runs" (68% confident)
6. **Continue** until predicting STOP signal

4.2 Three Key Mechanisms

1. Word Embeddings: Turning Words into Numbers

- Each word becomes 256 numbers
- Similar words get similar numbers

Example: "dog" might be [0.8, 0.2, 0.1, ...]. "puppy" might be [0.75, 0.25, 0.15, ...]. They're close because they're similar concepts. "car" would be very different: [-0.3, 0.9, -0.5, ...]

2. Position Markers: Remembering Word Order

- "dog runs" is different from "runs dog"
- We add position information to each word's numbers

Note: Think of it like page numbers in a book. Even if pages get shuffled, you can put them back in order using the page numbers.

3. Attention: Looking at What Matters

This is the secret sauce! At each step, the model decides:

- "Which previous words should I focus on?"
- "Which parts of the image should I look at?"

Example: When writing "runs" after "a dog", the model:

- Pays attention to "dog" (the subject)
- Ignores "a" (less important)
- Looks at image regions with motion

Key Idea: Attention is like having a spotlight. Instead of looking at everything equally, you shine the light on what's important for the current word.

4.3 The Architecture: 3 Layers of Processing

Each of our 3 layers does the same thing:

1. **Look at previous words** (self-attention)
2. **Look at the image** (cross-attention)
3. **Think about it** (process the information)

Note: Why 3 layers? Each layer refines the understanding:

- Layer 1: "I see a dog and grass"
- Layer 2: "The dog is moving on the grass"
- Layer 3: "It's running, not walking or sitting"

We tried 6 layers (model v5) but it was overkill for 8,000 images—like using a magnifying glass when you don't need one.

4.4 Why 4 Attention Heads?

Think of having 4 pairs of glasses, each helping you see different things:

- **Glasses 1:** Focuses on nouns (dog, grass, sky)
- **Glasses 2:** Focuses on verbs (running, playing, jumping)
- **Glasses 3:** Focuses on colors and details (brown, green, small)
- **Glasses 4:** Focuses on relationships (dog ON grass, dog CHASING ball)

Key Idea: Multiple heads let the model look at the same information from different perspectives simultaneously.

5 Training: Teaching the Model

5.1 The Learning Process

1. **Show photo + correct caption:** "a dog runs on grass"
2. **Model tries to generate caption:** "a dog walks on field"
3. **Calculate mistake:**
   - "walks" vs "runs": wrong action
   - "field" vs "grass": close but not exact
4. **Adjust numbers:** Make tiny changes so next time it's slightly better
5. **Repeat 32,000+ times** with different photos

5.2 The Two-Phase Training Strategy

Phase 1 (Epochs 1-3): Train Only the Writer

- The Observer (encoder) is frozen—its numbers don't change
- Only the Writer (decoder) learns

Key Idea: Why? The Observer already knows how to see (pre-trained on ImageNet). If we change it immediately, we might break this knowledge. First, we teach the Writer how to use the Observer's notes.

Example: It's like learning to cook from a professional chef. First, you learn to follow their instructions exactly. Only later do you start suggesting modifications to the recipe.

Phase 2 (Epochs 4-20): Fine-tune Everything

- Now both Observer and Writer learn together
- Learning rate reduced (smaller adjustment steps)

Note: We use smaller steps because the Observer's knowledge is valuable. We want to adjust it gently, not drastically change it.

5.3 Key Training Tricks

1. Label Smoothing: Don't Be Overconfident

Instead of:

- Next word is "dog": 100% sure

We teach:

- Next word is "dog": 99% sure
- Could be "puppy": 0.5% possible
- Could be "animal": 0.3% possible
- Others: 0.2% total

Key Idea: This prevents overconfidence. In real life, there are often multiple correct answers, so the model should allow for some uncertainty.

2. Data Augmentation: See Variations

During training, we randomly modify images:

- Flip horizontally (dog facing left $\leftrightarrow$ dog facing right)
- Change brightness (sunny day $\leftrightarrow$ cloudy day)
- Rotate slightly
- Convert to grayscale occasionally

Example: If we only show perfect, centered, well-lit photos during training, the model fails on blurry or dark test photos. Augmentation makes it robust.

3. Dropout: Forced Robustness

During training, randomly turn off 30% of the model's neurons.

Note: This is like practicing basketball with weights on your ankles. When you remove the weights (no dropout during testing), you perform even better.

4. Early Stopping: Know When to Stop

We stop training if the model doesn't improve for 5 epochs in a row.

Key Idea: Once the model stops learning, continuing to train just makes it memorize training data (overfitting). Better to stop early.

6 How We Measure Success

6.1 Validation Loss: The Practice Test

During training, we test on images the model has never seen.

- **High loss (e.g., 5.0):** Model is guessing randomly
- **Medium loss (e.g., 2.5):** Model is learning patterns
- **Low loss (e.g., 2.0):** Model is doing well

Our v6 result: 2.0972 (good!)

6.2 BLEU Score: Comparing to Human Captions

BLEU measures: "How similar is our caption to what humans wrote?"

The concept:

- Count how many words match
- Count how many word pairs match
- Count how many 4-word phrases match

Example:

Generated: "a dog runs on grass"
Human 1: "a brown dog running on grass"
Human 2: "dog runs through green field"

BLEU-1 (individual words):

- Matches: a, dog, runs, on, grass (5 out of 5)
- Score: $5/5 = 1.0$ (100%)

BLEU-4 (4-word phrases):

- Our 4-word phrases: "a dog runs on", "dog runs on grass"
- Human phrases: "a brown dog running", "brown dog running on", "dog running on grass", etc.
- Matches: None exactly
- Score: 0.0 (0%)

Note: BLEU-4 is much harder! It requires exact phrasing, not just using the right words.

Our v6 results:

- **BLEU-1: 0.57** — We use correct words 57% of the time
- **BLEU-4: 0.10** — We match exact phrases 10% of the time

6.3 What's Good Enough?

Model Type	BLEU-1	BLEU-4
Random guessing	0.05	0.00
Basic CNN-LSTM	0.55	0.20
Our v6	**0.57**	**0.10**
Attention models	0.63	0.25
Transformer (large)	0.67	0.28

Key Idea: We beat the basic baseline and are approaching more complex models. For 8,000 training images, this is solid performance!

7 Why v6 Beats v5 and v7

7.1 The Goldilocks Problem

Model v5 (33 million parameters):

- **Problem:** Too big for 8,000 images
- **What happened:** Memorized training photos
- **Result:** Good on training, bad on testing (overfitting)

Example: Like studying for a test by memorizing answers without understanding concepts. You ace practice problems but fail on new questions.

Model v6 (9 million parameters):

- **Size:** Just right for 8,000 images
- **What happened:** Learned general patterns
- **Result:** Best performance on testing

Model v7 (16 million parameters):

- **Problem:** Too conservative
- **What happened:** Played it safe with generic captions
- **Result:** Low loss but boring captions

Example: Like writing "The image shows a scene" for every photo. Technically correct (low loss) but useless (low BLEU).

7.2 The Evidence

Training vs Testing Gap:

Model	Gap	Meaning
v5	+0.5	Training much better → overfitting
v6	-0.15	Slight negative → perfect!
v7	-0.25	Too negative → too safe

Note: Small negative gap is ideal because:

1. Training is harder (we use augmentation, dropout)
2. Testing is easier (clean images, no randomness)
3. Model generalizes well

8 Generating Captions: Beam Search

When generating captions, we don't just pick the most likely word each time. We use **beam search**.

8.1 The Problem with Greedy Search

Greedy approach:

- Pick most likely first word: "the" (40%)
- Pick most likely second word: "dog" (50%)
- Result: "the dog ..."

Problem: What if "a dog" (35% + 60%) is actually better overall than "the dog" (40% + 50%)?

8.2 Beam Search: Keep Options Open

Instead of committing to one path, we explore multiple possibilities:

1. Start: `<START>`
2. Generate top 5 first words:
   - "a" (35%)
   - "the" (30%)
   - "two" (15%)
   - "several" (10%)
   - "some" (5%)
3. For each, generate top 5 next words (25 total sequences)
4. Keep only top 5 sequences by total probability
5. Repeat until complete

Example: It's like playing chess: instead of committing to your first move immediately, you think several moves ahead and choose the path that looks best overall.

Key Idea: We use beam width = 5. Bigger (10, 20) is slower but better. Smaller (1, 2) is faster but worse. 5 is the sweet spot.

9 Complete Pipeline Summary

9.1 Training (Learning)

1. Get photo + correct caption
2. Observer looks at photo → 256 numbers
3. Writer reads numbers + generates caption word-by-word
4. Compare to correct caption
5. Adjust model to reduce mistakes
6. Repeat 32,360 times

9.2 Testing (Using)

1. Get new photo
2. Observer looks at photo → 256 numbers
3. Writer generates caption using beam search
4. Output: "a dog runs on grass"

10 Key Takeaways

1. Two-Part Design:

- Encoder (Observer) = Pre-trained expert who understands images
- Decoder (Writer) = Learned to turn image features into words

2. Size Matters:

- v5 too big → memorization
- v6 just right → learning
- v7 bigger but worse → too conservative

3. Smart Training:

- Use expert knowledge (pre-trained encoder)
- Train in phases (freeze then unfreeze)
- Prevent overconfidence (label smoothing, dropout)
- Stop at the right time (early stopping)

4. Generation Strategy:

- Beam search explores multiple options
- Better than greedy but faster than exhaustive

5. Success Metrics:

- BLEU-1 measures word accuracy (57%)
- BLEU-4 measures phrase accuracy (10%)
- Both together show we're learning meaningful patterns

Final thought: Model v6 succeeded because it's the right size for the job. Not too smart (memorizing), not too simple (generic), but just right for learning patterns from 8,000 images.