Understanding the workload (2)

Aaron Zhao, Imperial College London

Introduction

Learning Objectives

By the end of this lecture, you should be able to:

- Understand how modern foundation models are constructed from basic building blocks
- Identify key architectural innovations in LLaMA2, CLIP, SAM, and Whisper
- Recognize the role of pre-training strategies (contrastive learning, masked autoencoding, etc.)
- Analyze how different modalities (text, image, audio) are encoded and aligned in multi-modal models

Introduction - Outline

Using the basic networks and building blocks we learned in the previous lecture, we will look at how researchers have constructed

- LLAMA2
- CLIP
- SAM
- Whisper

GPT's training detail is not open-sourced, and it is (or should be) somehow very similar to the OPT and LLaMA models that we have looked at.

LLaMA2

Introduction - Outline (iii)

LLaMA normally refers to the LLaMA model, LLaMA-Chat refers to the chatbot.

What is the difference?

- LLaMA is fully trained with only the pre-training data.
- LLaMA-Chat requires additional treatment such as an iterative refinement using Reinforcement Learning with Human Feedback (RLHF).

This is almost the same for all state-of-the-art LLMs! There are additional steps required to make them a good chatbot!

LLaMA2 pre-training details

Architecture details

- Standard Transformer decoder-only architecture
- Pre-normalization with RMSNorm
- Rotary positional embedding (RoPE)
- Grouped-query attention (GQA)
- SwiGLU activation function

LLaMA2 Model Sizes

Table 1: LLaMA2 model variants with different sizes

Model	Parameters	Layers	Hidden Dim	Attention Heads
LLaMA2-7B	7 billion	32	4096	32
LLaMA2-13B	13 billion	40	5120	40
LLaMA2-70B	70 billion	80	8192	64 (8 KV)

- All models use context length of 4096 tokens (2× from LLaMA1)
- 70B model uses Grouped-Query Attention (8 KV heads for 64 query heads)

Grouped-Query Attention (GQA)

GQA is an optimization that balances quality and inference speed by sharing key-value heads across groups of query heads.

Multi-Head Attention (MHA):

- Each query head has its own key and value heads
- H query heads, H key heads, H value heads
- Most expressive but memory-intensive

Multi-Query Attention (MQA):

- All query heads share single key and value heads
- H query heads, 1 key head, 1 value head
- Fastest but may reduce quality

Grouped-Query Attention (GQA):

- Middle ground
- H query heads, G key heads, G value heads (where $1 < G < H$)
- LLaMA2-70B: 64 query heads, 8 key-value heads (8 queries per group)

Grouped-Query Attention (GQA) (ii)

Benefits:

- Reduces KV cache size during inference (critical for long contexts)
- Near MHA quality with near MQA speed
- Enables larger batch sizes and higher throughput

Training details

- Two trillion tokens training data!
- AdamW optimizer
- Cosine learning rate scheduler with warmup

Normalization

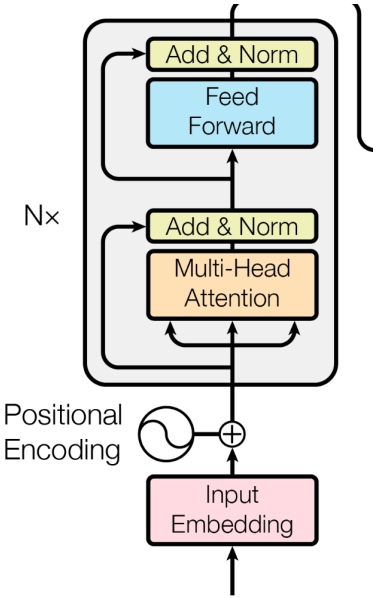
- Pre- and Post-Normalization
- Post-Normalization

$$x = f_N(x + f(x))$$

- Pre-Normalization

$$x = x + f(f_N(x))$$

Normalization (ii)



LayerNorm and RMSNorm

LayerNorm: $y = \frac{x-\mu}{\sigma}\alpha$

μ is the mean and σ is the std, α is a learnable gain parameter.

RMSNorm: $y = \frac{x}{RMS(x)}\alpha$

$RMS(x) = \sqrt{\frac{1}{n}\sum_{i=0}^{n-1} x_i^2}$

Why RMSNorm is more efficient:

- No mean computation (removes μ calculation)
- No variance computation (only RMS needed)
- 10-15% faster than LayerNorm
- Empirically similar performance in LLMs

LayerNorm requires: mean $\longrightarrow$ variance $\longrightarrow$ normalize

RMSNorm requires: RMS $\longrightarrow$ normalize (fewer operations!)

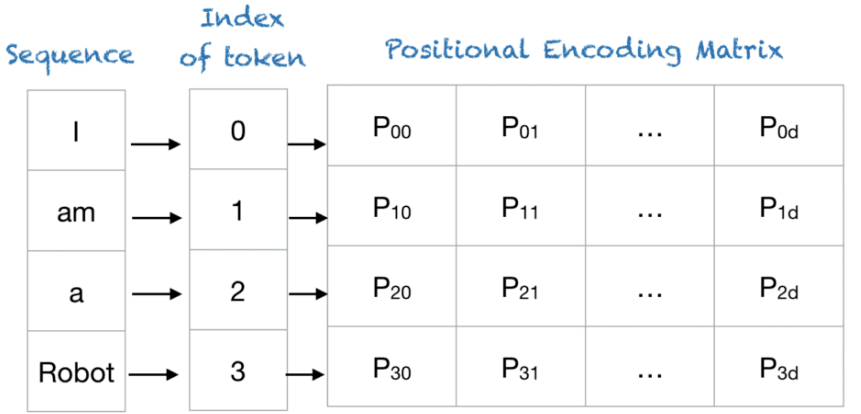
Positional Embedding

The original input is a sequence of word embedding $X_e \in \mathcal{R}^{N \times D}$.

Classical positional embedding $X_p \in \mathcal{R}^N$ is added to the word embedding to provide $X = X_e + X_p$.

Without positional embedding, the model cannot tell the difference between "I am a robot" and "am I a robot", because we take these inputs in parallel.

Positional Embedding (ii)



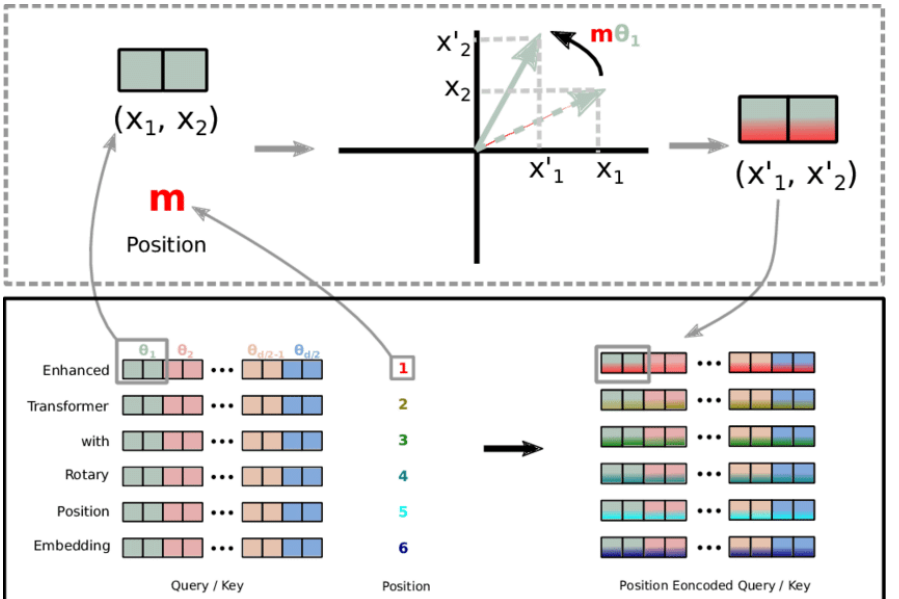
Positional Encoding Matrix for the sequence 'I am a robot'

Rotary Positional Embedding (RoPE)

- Absolute positional embedding: sinusoidal functions (no length constraints)
- Relative positional embedding: additional components added to the Q, K components, such as T5, slow at large sequences, no KV caching, not commonly used in large models.
- Rotary positional embedding (RoPE): proposed in RoFormer, derived from constraints of the attention mechanism (inner product between query and key) and relative positional encoding ($(m-n)$).

$$\langle f_{q(\boldsymbol{x}_m, m)}, f_{k(\boldsymbol{x}_n, n)} \rangle = g(\boldsymbol{x}_m, \boldsymbol{x}_n, m-n)$$

Rotary Positional Embedding (RoPE) (ii)



SwiGLU Activation Function

LLaMA2 uses SwiGLU instead of ReLU or GELU in the feed-forward network.

Traditional Activations:

- ReLU
  - $\text{ReLU}(x) = \max(0, x)$ - simple but non-smooth at zero
- GELU (Gaussian Error Linear Unit): $\text{GELU}(x) = x \cdot \Phi(x)$
  - where $\Phi(x)$ is the cumulative distribution function of standard normal
  - Approximation: $\text{GELU}(x) \approx 0.5x\left(1 + \tanh\left(\sqrt{\frac{2}{\pi}}(x + 0.044715x^3)\right)\right)$
  - Smooth, non-monotonic (allows small negative values)
  - Used in BERT, GPT-2, GPT-3

SwiGLU (Swish-Gated Linear Unit):

$$\text{SwiGLU}(x, W, V, b, c) = \text{Swish}(xW + b) \circ (xV + c)$$

where $\text{Swish}(x) = x \cdot \sigma(x)$ (also called SiLU) and $\circ$ is element-wise multiplication.

SwiGLU Activation Function (ii)

Key Properties:

- Gating mechanism: one linear projection gates another
- Smoother than ReLU, similar to GELU but with gating
- Better empirical performance in large models (PaLM, LLaMA, Mistral)
- Slightly more computation (50% more FLOPs) but worth it for quality

Contrastive Language–Image Pre-training (CLIP)

CLIP

Classic computer vision systems (such as ResNets) have the following disadvantages:

- Trained on labelled data.
- Normally predict a fixed set of predetermined object categories
- New labelled data is needed if you want to specify any other visual concept

Can we learn directly from raw text about images and the images?

The core idea is to use natural language as supervision.

CLIP (ii)

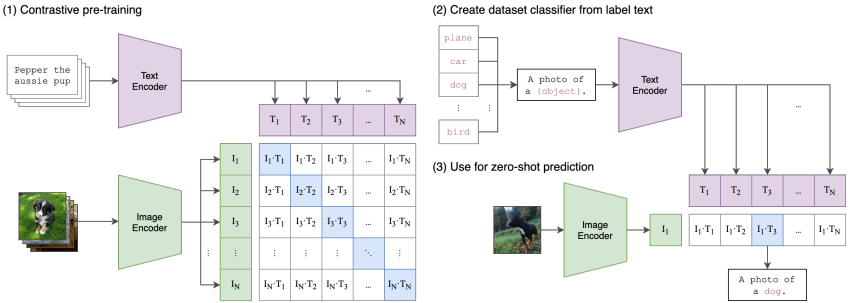


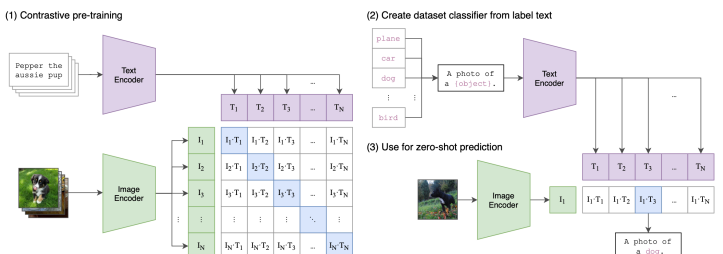
Figure 4: CLIP training approach: jointly training image and text encoders with contrastive learning

Contrastive representation learning works by predicting which of the $N \times N$ possible (image, text) pairings across a batch actually occurred.

CLIP (iii)

This forces learning in a multi-modal embedding space by jointly training an image encoder and text encoder to

- maximize the cosine similarity of the image and text embeddings of the N real pairs in the batch
- minimize the cosine similarity of the embeddings of the $N^2 - N$ incorrect pairings.



CLIP Contrastive Loss

Given a batch of N (image, text) pairs with embeddings $\boldsymbol{I} \in \mathcal{R}^{N \times d}$ and $\boldsymbol{T} \in \mathcal{R}^{N \times d}$:

1. Compute pairwise cosine similarities: $S = \frac{\boldsymbol{I}\boldsymbol{T}^T}{\tau}$

   where τ is a learned temperature parameter

2. Symmetric cross-entropy loss:

$$\mathcal{L} = \frac{1}{2}(\mathcal{L}_{I \to T} + \mathcal{L}_{T \to I})$$

$$\mathcal{L}_{I \to T} = -\frac{1}{N} \sum_{i=1}^{N} \log \frac{\exp(S_{ii})}{\sum_{j=1}^{N} \exp(S_{ij})}$$

Key Insight: Train on 400M image-text pairs from the web, no manual labels needed!

CLIP Zero-Shot Classification

CLIP enables zero-shot classification without training on specific categories!

Process:

1. **Create text prompts** for each class:
   - Instead of just “cat”, use “a photo of a cat”
   - Better: prompt ensemble with templates like “a photo of a {class}”, “a picture of a {class}”
2. **Encode everything:**
   - Image $\longrightarrow$ image encoder $\longrightarrow \boldsymbol{v}_{\text{img}} \in \mathcal{R}^d$
   - Each text prompt $\longrightarrow$ text encoder $\longrightarrow \boldsymbol{v}^i_{\text{text}} \in \mathcal{R}^d$ for $i = 1, ..., K$ classes
3. **Compute similarities:**

$$p\big(y = i \mid \boldsymbol{v}_{\text{img}}\big) = \frac{\exp\left(\frac{\text{sim}\big(\boldsymbol{v}_{\text{img}}, \boldsymbol{v}^i_{\text{text}}\big)}{\tau}\right)}{\sum_{j=1}^{K} \exp\left(\frac{\text{sim}\big(\boldsymbol{v}_{\text{img}}, \boldsymbol{v}^j_{\text{text}}\big)}{\tau}\right)}$$

CLIP Zero-Shot Classification (ii)

4. **Predict:** Choose class with highest probability

Why it works: The shared embedding space learned during contrastive pre-training aligns images with their textual descriptions!

We can also fine-tune CLIP by connecting the image encoder with a task-specific classifier.

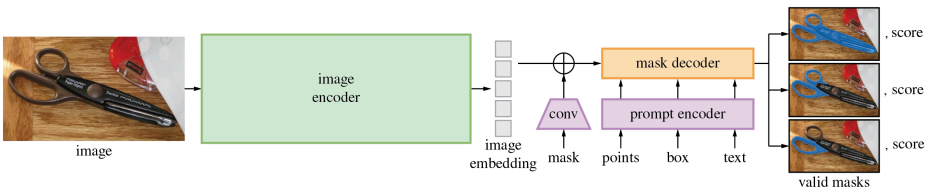
CLIP (contrastive learning) opened the door for building **vision-language foundation models**.

Segment Anything Model (SAM)

Segment Anything

A foundation model for image segmentation

- A large-scale dataset on the task: 1+ billion masks and 11 million images
- Three-component architecture working together



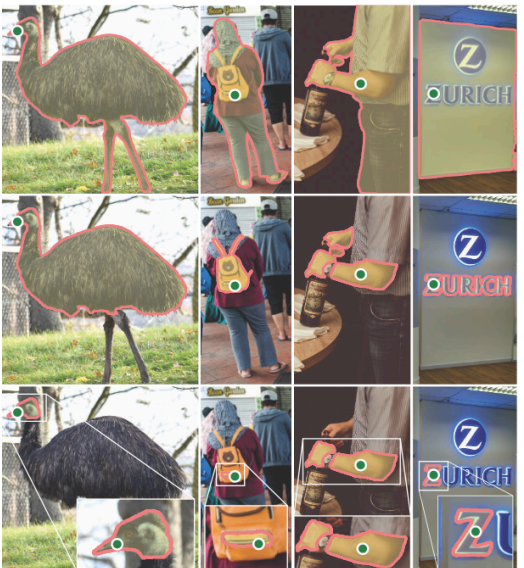
SAM model architecture: image encoder, prompt encoder, and mask decoder working together

SAM Architecture: How Components Work

1. **Image Encoder** (one-time, heavy computation)
   - Input: RGB image 1024×1024
   - MAE-pretrained ViT-H (Huge) backbone
   - Output: Image embedding $64 \times 64 \times 256$
   - Run once per image, reused for multiple prompts!
2. **Prompt Encoder** (lightweight, per-prompt)
   - Takes user prompts as input
   - Outputs prompt embeddings
   - Runs for each new prompt
3. **Mask Decoder** (lightweight, per-prompt)
   - Combines image embedding + prompt embeddings
   - Predicts segmentation mask(s)

Key Design: Heavy computation (image encoding) done once, then fast interactive prompting!

Segment Anything



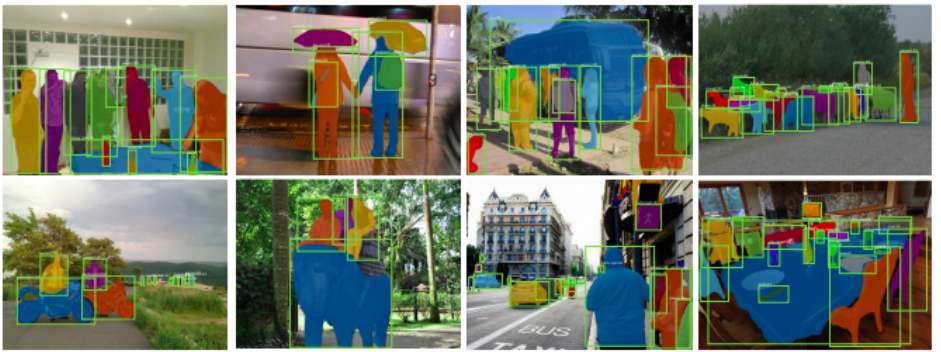
Segmentation tasks

Image segmentation is a computer vision and image processing technique that involves grouping or labeling similar regions or segments in an image on a pixel level.

- Semantic segmentation: Segments amorphous regions (or repeating patterns) of similar material, which is uncountable (e.g., road, sky, and grass).
- Instance segmentation: Segments countable objects in an image (e.g., people, flowers, birds, animals, etc.).
- Panoptic segmentation: Combines both.

Segmentation tasks (ii)

Segmentation tasks (iii)



Segmentation tasks (iv)

Figure 10: Panoptic Segmentation

SAM Model Architecture

- Image Encoder
- Prompt Encoder
- Mask Decoder

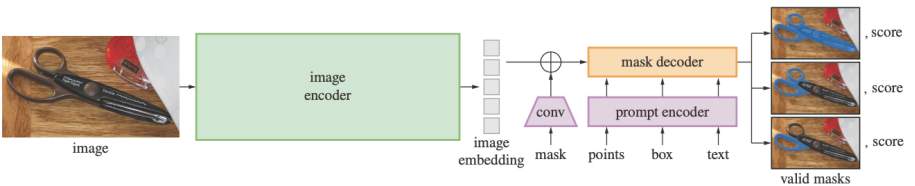
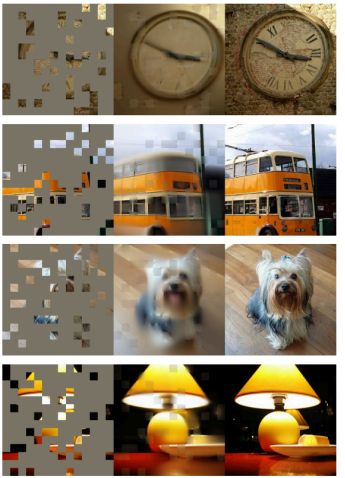


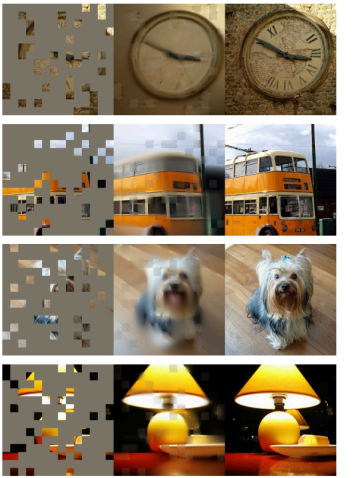
Image Encoder in SAM

- Original Image in shape of 1024×1024
- Pretrained **Masked AutoEncoder**
- Transform to embedding $64 \times 64 \times 256$

SAM Model Architecture (ii)

Mask AutoEncoder (MAE)





Masked AutoEncoder (MAE) Details

MAE is a self-supervised learning method that pre-trains Vision Transformers.

Key Idea: Mask large portions of the image and reconstruct the missing pixels.

Architecture:

1. **Input:** Divide image into patches (e.g., $224 \times 224 \longrightarrow 14 \times 14$ patches of 16×16)
2. **Masking:** Randomly mask 75% of patches (much higher than BERT's 15%!)
3. **Encoder:** ViT processes only visible patches (25%)
   - No mask tokens in encoder (saves computation!)
   - Produces embeddings for visible patches only
4. **Decoder:** Lightweight transformer reconstructs all patches
   - Takes visible patch embeddings + learned mask tokens

Masked AutoEncoder (MAE) Details (ii)

- Predicts pixel values for masked patches

5. **Loss:** MSE between reconstructed and original pixels (only on masked patches)

$$\mathcal{L}_{\text{MAE}} = \frac{1}{M} \sum_{\text{masked patches}} \| \boldsymbol{x}_{\text{pred}} - \boldsymbol{x}_{\text{true}} \|^2$$

Why it works for SAM: Forces encoder to learn rich semantic representations that generalize well to segmentation!

Handling Different Prompt Types

SAM supports multiple prompt types, all encoded into a common embedding space:

1. Points (foreground/background clicks)

- Represent each point as (x, y, type)
- Positional encoding using learned spatial frequencies
- Add learned embedding for foreground vs background
- Output: Sparse prompt embeddings

2. Bounding Boxes

- Represented as 2 corner points
- Each corner encoded like a point
- Learns that these define a rectangular region

3. Text Prompts (e.g., "segment the dog")

- Use CLIP text encoder
- Projects text into SAM's embedding space

Handling Different Prompt Types (ii)

- Enables language-guided segmentation

4. Mask Prompts (rough mask from previous iteration)

- Convolve with downsampling (to match 64×64 resolution)
- Extract dense embeddings
- Useful for iterative refinement

All prompt types $\longrightarrow$ unified embedding space $\longrightarrow$ fed to mask decoder!

Mask Decoder: Predicting Segmentation Masks

The mask decoder is a modified transformer that fuses prompts with image embeddings:

Architecture:

1. **Inputs:**
   - Image embedding: $64 \times 64 \times 256$
   - Prompt embeddings (sparse or dense)
   - Learnable output tokens (for mask + IoU)
2. **Two-way transformer blocks** (2 layers):
   - Self-attention on prompt tokens (prompts attend to each other)
   - Cross-attention: prompts $\longrightarrow$ image (update prompts with image info)
   - Cross-attention: image $\longrightarrow$ prompts (update image with prompt info)
   - This bidirectional flow creates prompt-aware image features!
3. **Upsampling & prediction:**
   - Upsample image embedding: $64 \times 64 \longrightarrow 256 \times 256$

Mask Decoder: Predicting Segmentation Masks (ii)

- MLP head predicts per-pixel mask logits
- Separate head predicts IoU score (quality estimate)

4. **Multiple masks:**
   - Predicts 3 candidate masks simultaneously
   - Each with its own IoU score
   - User/system picks best one based on IoU

Output: Segmentation mask + confidence score!

SAM Inference Example: Step-by-Step

Scenario: Segment a cat in an image

Step 1: Encode image (once)

- Input: Photo with cat $(1024 \times 1024 \times 3)$
- Image encoder: ViT-H processes image
- Output: Image embedding $(64 \times 64 \times 256)$
- ⏱️ Slowest step (0.5 seconds), but only once!

Step 2: User provides prompt

- Option A: Click on cat's head (foreground point)
- Option B: Click box around cat
- Option C: Type "segment the cat"

Step 3: Encode prompt (fast)

- Prompt encoder converts click/box/text $\longrightarrow$ embeddings
- ⏱️ Very fast (0.001 seconds)

SAM Inference Example: Step-by-Step (ii)

Step 4: Decode mask (fast)

- Mask decoder fuses image embedding + prompt embeddings
- Predicts 3 candidate masks with IoU scores
- ⏱️ Fast (0.01 seconds)

Step 5: Refinement (optional, interactive)

- If mask not perfect, user adds more clicks
- Decoder reuses same image embedding
- ⏱️ Each iteration 0.01 seconds (interactive!)

Total: First prompt 0.5s, subsequent prompts 0.01s each!

SAM Inference Example: Step-by-Step (iii)

The Most Important Part: SA-1B dataset

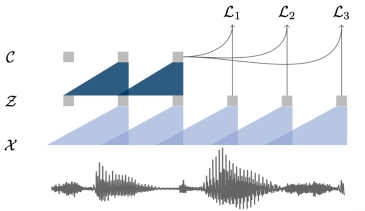
Whisper

Whisper

Speech recognition is a task of converting spoken language to text. It involves recognizing the words spoken in an audio recording and transcribing them into a written format (text).

WAV2Vec The ‘classic’ speech recognition network

WAV2Vec



- An encoder network: takes audio signal as inputs and project them to an embedding space using Convolutions
- A context network: combines multiple time-steps of the encoder to obtain contextualized representations also using Convolutions.
- Limited context length.

Whisper Architecture

Multitask training data (680k hours)

English transcription

- "Ask not what your country can do for …"
- Ask not what your country can do for …

Any-to-English speech translation

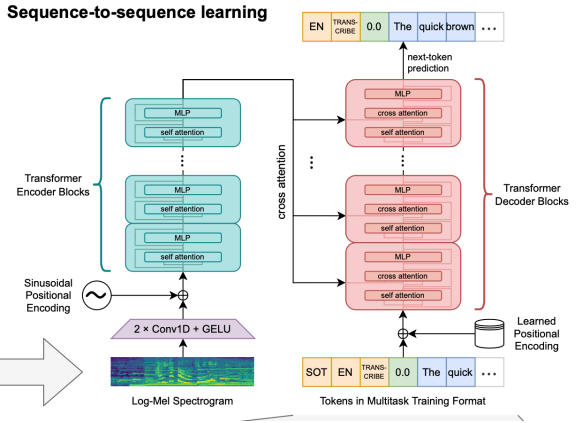
- "El rápido zorro marrón salta sobre …"
- The quick brown fox jumps over …

Non-English transcription

- "언덕 위에 올라 내려다보면 너무나 넓고 넓은 …"
- 언덕 위에 올라 내려다보면 너무나 넓고 넓은 …

No speech

- (background music playing)
- ∅



log-Mel Spectrogram

- Transform signal from the time domain to frequency domain
- Studies have shown that humans do not perceive frequencies on a linear scale. We are better at detecting differences in lower frequencies than higher frequencies.
  - we can easily tell the difference between 500 and 1000 Hz
  - but we will hardly be able to tell a difference between 10,000 and 10,500 Hz
- In 1937, Stevens, Volkmann, and Newmann proposed a unit of pitch such that equal distances in pitch sounded equally distant to the listener. This is called the mel scale.
- Perform a unit conversion to the log-mel scale
- All audio is re-sampled to 16,000 Hz, and an 80-channel log-magnitude Mel spectrogram representation is computed on 25-millisecond windows with a stride of 10 milliseconds.

log-Mel Spectrogram (ii)

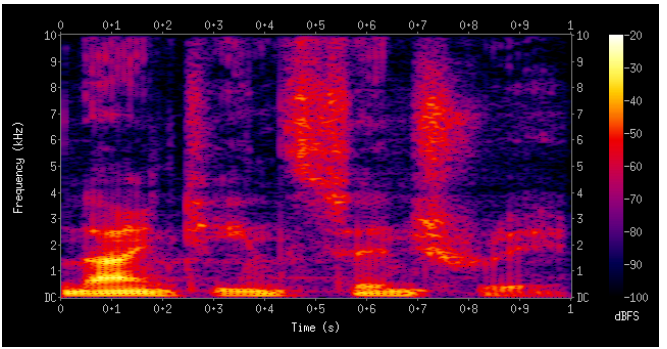


Figure 16: Mel spectrogram visualization: time on x-axis, frequency on y-axis, intensity shown by color

Whisper Architecture Details

Input Processing:

- Audio resampled to 16 kHz
- Conv1D layers (kernel size 3, stride 2) reduce sequence length
  - From 3000 frames $\longrightarrow$ 1500 features
  - GELU activation
  - Creates valid embedding for transformer

Encoder-Decoder:

- Classic Transformer with self-attention and cross-attention
- Sinusoidal positional embeddings
- Decoder uses special tokens to control behavior

Whisper Multi-Task Training

Whisper is trained on 680,000 hours of multilingual and multitask data to perform:

Task Types:

1. **Transcription:** Audio → text in same language
2. **Translation:** Audio → English text (X-to-English)
3. **Language Detection:** Identify spoken language
4. **Voice Activity Detection:** Detect speech presence
5. **Timestamp Prediction:** Word-level timing

Special Tokens Control Tasks:

- `<|startoftranscript|>` - Begin decoding
- `<|en|>`, `<|zh|>`, etc. - Language codes
- `<|translate|>` or `<|transcribe|>` - Task selection
- `<|notimestamps|>` - Enable/disable timestamps

Whisper Multi-Task Training (ii)

Example: `<|startoftranscript|><|en|><|transcribe|><|notimestamps|>` "Hello world"

Training Strategy: All tasks trained jointly, model learns to switch based on tokens!

Summary

- Introduced three new tasks:
  - Unsupervised learning (contrastive learning)
  - Image segmentation
  - Speech Recognition
- A few model architectures (CLIP, LLaMA2, SAM, Whisper)
- Reuse classic and powerful building blocks (eg. self-attention, image encoder)

Model Architecture Comparison

Table 2: Comparison of foundation model architectures covered in this lecture

Model	**Primary Task**	**Key Innovation**	**Encoder/ Decoder**	**Training Strategy**
LLaMA2	Language Modeling	RMSNorm, RoPE, GQA	Decoder-only	Causal LM on 2T tokens
CLIP	Vision-Language	Contrastive Learning	Dual Encoders	Image-text pair matching
SAM	Segmentation	Promptable Masks	Encoder + Decoder	SA-1B dataset (1B+ masks)
Whisper	Speech Recognition	log-Mel Spectrogram	Encoder-Decoder	680k hours audio data

Key Insights: Foundation Model Design Patterns

LLaMA2:

- Efficiency improvements: RMSNorm reduces computation vs LayerNorm
- RoPE enables better length generalization
- GQA reduces KV cache size for longer contexts

CLIP:

- Multi-modal alignment through contrastive learning
- Zero-shot transfer: no task-specific fine-tuning needed
- Scales with web-scraped image-text pairs

SAM:

- Promptable design: points, boxes, text, or masks as prompts
- MAE pre-training for robust image embeddings

Whisper:

- Mel scale mimics human auditory perception
- Encoder-decoder for flexible output (transcription, translation)

Common Themes Across Models

1. **Pre-training at Scale**: All models benefit from large-scale pre-training
   - LLaMA2: 2 trillion tokens
   - CLIP: 400 million image-text pairs
   - SAM: 1+ billion masks
   - Whisper: 680k hours of audio
2. **Transfer Learning**: Pre-trained models adapt to downstream tasks
3. **Architectural Efficiency**: Innovations reduce computational cost
   - RMSNorm, GQA (LLaMA2)
   - Lightweight mask decoder (SAM)
   - Conv1D preprocessing (Whisper)
4. **Multi-modal Alignment**: Bridging different data types
   - CLIP: vision $\leftrightarrow$ language
   - SAM: image $\leftrightarrow$ prompts
   - Whisper: audio $\longrightarrow$ text

Further Reading

LLaMA2:

- "LLaMA 2: Open Foundation and Fine-Tuned Chat Models" (Meta AI, 2023)
- Cameron R. Wolfe's "LLaMA-2 from the Ground Up"

CLIP:

- "Learning Transferable Visual Models From Natural Language Supervision" (OpenAI, 2021)
- CLIP GitHub repository: github.com/openai/CLIP

SAM:

- "Segment Anything" (Meta AI, 2023)
- SA-1B Dataset paper

Whisper:

- "Robust Speech Recognition via Large-Scale Weak Supervision" (OpenAI, 2022)
- Whisper GitHub repository: github.com/openai/whisper