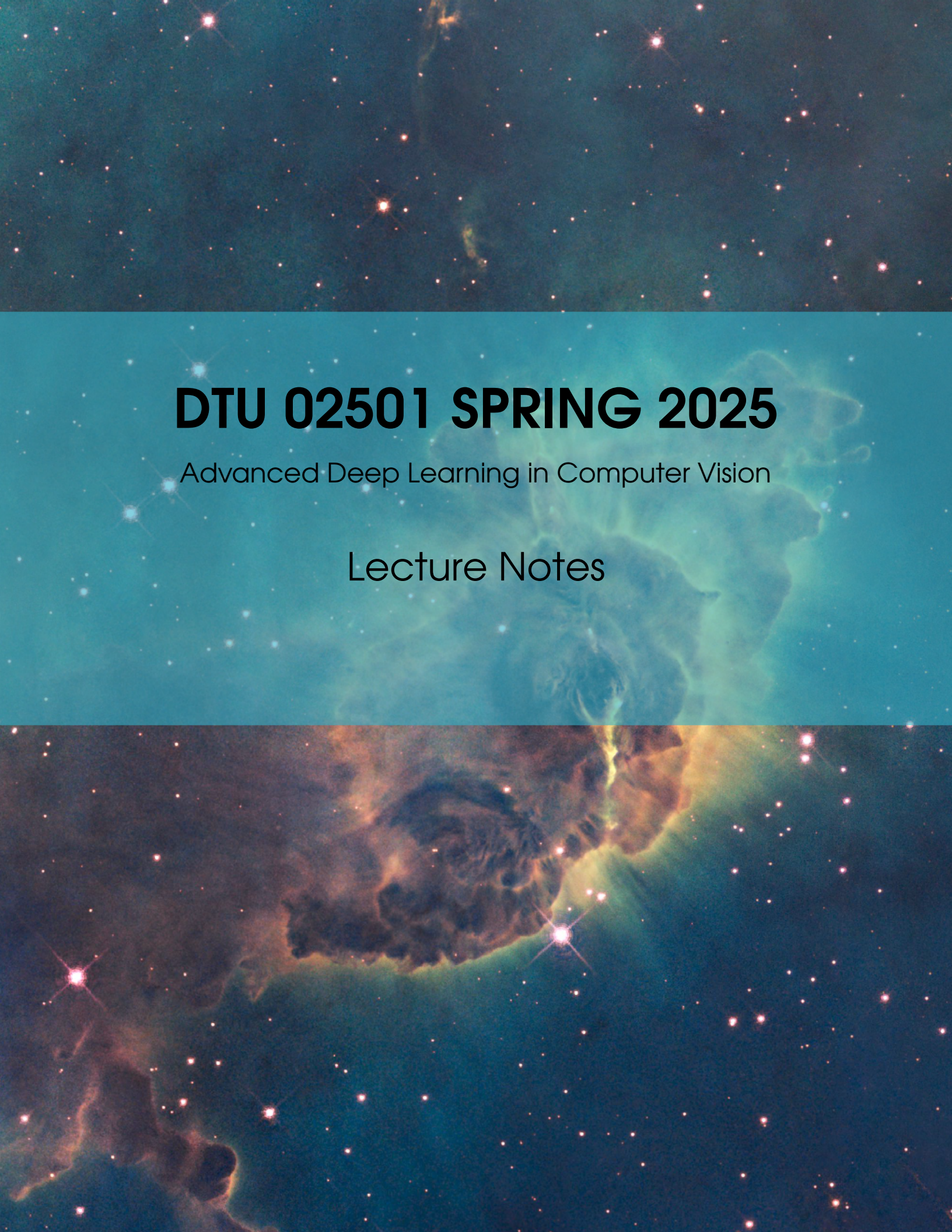


DTU 02501 SPRING 2025

Advanced Deep Learning in Computer Vision

Lecture Notes

Contents

1 Lecture 1: Transformers, Attention and Encoders 7

1.1 Sequence-to-Sequence Models & Limitations of RNNs 7

1.2 Transformer Architecture 7

1.3 Self-Attention Mechanism 8

1.4 Multi-Head Attention 8

1.5 Positional Encoding 8

1.6 Transformer for Classification 9

1.7 BERT Overview 9

2 Lecture 2: Visual Transformers 11

2.1 Foundational Concepts 11

2.2 Vision Transformer (ViT) Architecture 11

2.3 Hierarchical Vision Transformers 12

2.4 Object Detection with Transformers 12

2.5 Image Segmentation with Transformers 13

2.6 Advanced Concepts 13

2.7 Quiz Questions with Answers 13

3 Lecture 3: Transformer Decoders, LLMs and GPTs 15

3.1 GPT-Style Models vs BERT 16

3.2 Machine Translation with Transformers 16
3.3 Key Diagrams & Workflows 17
3.4 Practice Quiz Questions 17
3.5 Common Pitfalls & Advanced Questions 18

4 Lecture notes: Diffusion Models 19
4.1 The Forward Process (Diffusion/Noise Addition) 19
4.2 The Reverse Process (Denoising) 20
4.3 Learning to Denoise 21
4.4 Training and Sampling Process 21
4.5 Key Exam Topics and Common Questions 23
4.6 Practice Questions for Exam Preparation 23
4.7 Common Pitfalls and Deep Insights 24

5 Lecture 4: Diffusion Models 25
5.1 Overview of Diffusion Models 25
5.2 The Forward and Reverse Processes 25
5.3 Model Architecture and Training 26
5.4 Important Visual Examples 27
5.5 Important Visual Examples 27
5.6 Quiz Questions and Answers 27
5.7 Details Often Overlooked 28
5.8 Additional Study Tips 28

6 Lecture 5: Diffusion Guidance 31
6.1 Guidance in Diffusion Models: Comprehensive Exam Preparation 31
6.2 Types of Guidance 31
6.2.1 2.1 Classifier Guidance 31
6.2.2 2.2 Classifier-Free Guidance 32
6.3 Universal Guidance 32
6.4 Counterfactual Explanations with Diffusion Guidance 32
6.5 Key Implementation Details and Challenges 33
6.6 Evaluation Metrics for Guided Diffusion 33
6.7 Oral Exam-Style Quiz Questions & Answers 33
6.8 Common Misconceptions and Subtleties 34

6.9 Advanced Applications and Recent Developments 34
6.10 Common Misconceptions and Subtleties 35
6.11 Advanced Applications and Recent Developments 35

7 Lecture 6: SOTA Text2Image Generation 37
7.1 Core Concepts of Latent Diffusion Models (LDMs) 37
7.2 Detailed Look at Latent Diffusion Model Architecture 38
7.3 CLIP: The Vision-Language Model Powering Text Conditioning 38
7.4 Text-to-Image Generation End-to-End (Stable Diffusion) 39
7.5 Key Visualizations and Examples 39
7.6 Exam-Style Questions with Answers 40
7.7 Advanced Topics Often Overlooked 41

8 Lecture 7: self-Supervised Learning 43
8.1 Why Self-Supervised Learning? 43

9 Lecture 8: Multi-Modal Learning 49

10 Lecture 9: Explainable AI 55

11 Lecture 10: Fairness 59

12 Exercise 1 63

13 Exercise 2 67

14 Exercise 3 AndersenGPT 71

15 Exercise 2 Guided Diffusion Models 75

16 Exercise 3: Textual Inversion 79

17 Stable Diffusion vs Stable Diffusion XL 83

18 Project Related in-depth knowledge 87
18.1 Problem 87
18.2 Solution 87

18.3 Data **87**
18.4 Data preprocessing **87**
18.5 Models **87**
18.5.1 Vision Language Models 87
18.5.2 Diffusion Models 87
18.5.3 Vision Transformer 87
18.6 Training **87**
18.6.1 Parameters 87
18.6.2 Loss Functions 87
18.7 Results **87**

19 Glossary of Terms 89
19.1 Explainability and Fairness **89**
19.2 Transformers and Language Models **89**
19.3 Generative Models and Diffusion **90**
19.4 Self-Supervised and Multimodal Learning **91**
19.5 Miscellaneous **92**

1. Lecture 1: Transformers, Attention and Encoders

1.1 Sequence-to-Sequence Models & Limitations of RNNs

RNNs limitations (critical to understand):

- Cannot be trained in parallel (sequential processing)
- Past information influence is quickly lost over time
- Difficult to train due to vanishing/exploding gradients
- Formula chain for backpropagation:

$$\frac{\partial h_t}{\partial h_0} = \left(\frac{\partial h_t}{\partial h_{t-1}}\right)\left(\frac{\partial h_{t-1}}{\partial h_{t-2}}\right)\cdots\left(\frac{\partial h_1}{\partial h_0}\right)$$

1.2 Transformer Architecture

Core innovation:

Relies entirely on attention mechanisms without recurrence or convolutions

Structure:

Encoder-Decoder architecture with multiple identical layers

Key components:

- Self-attention mechanism
- Multi-head attention
- Positional encoding
- Feed-forward networks
- Add & Norm (residual connections + layer normalization)

1.3 Self-Attention Mechanism

Key equation:

$$\text{Attention}(Q,K,V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

Components:

- **Q (Query)**: What we're looking for
- **K (Key)**: What we match against
- **V (Value)**: What we retrieve

Properties:

Permutation-equivariant (needs positional encoding for order)

■ **Example 1.1** *"The animal didn't cross the street because it was too tired,"* self-attention helps **"it"** attend to **"animal"**

■

1.4 Multi-Head Attention

Purpose:

Allows the model to jointly attend to information from different representation subspaces

Process:

1. Project input into multiple sets of Q, K, V
2. Perform attention on each set independently
3. Concatenate results and project back to original dimension

Formula:

$$\text{MultiHead}(Q,K,V) = \text{Concat}(\text{head}_1,\ldots,\text{head}_h)W^O$$

$$\text{where head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

1.5 Positional Encoding

Purpose:

Add sequence order information since transformer processes tokens in parallel

Original implementation:

- $\text{PE(pos, 2i)} = \sin\left(\frac{\text{pos}}{10000^{\frac{2i}{d_{\text{model}}}}}\right)$
- $\text{PE(pos, 2i+1)} = \cos\left(\frac{\text{pos}}{10000^{\frac{2i}{d_{\text{model}}}}}\right)$

Alternative:

Learnable positional encodings (`nn.Parameter`)

Application:

Added to input embeddings before feeding to the model

1.6 Transformer for Classification

Architecture modifications:

1. Remove Transformer Decoder
2. Add pooling at the Encoder output (avgpool or [CLS] token)
3. Add a simple Linear Layer as “decoder”

(CLS) token approach:

Special token that aggregates sequence information through self-attention

1.7 BERT Overview

- **Full name:** Bidirectional Encoder Representations from Transformers
- **Pre-training tasks:**
  - Masked Language Modeling (MLM)
  - Next Sentence Prediction (NSP)
- **Fine-tuning:** Add task-specific layers for classification, NER, etc.

Visual Explanations & Examples to Remember

1. **YouTube Search as Q-K-V analogy:**
   - Query: What you type in search bar
   - Keys: Video metadata in database
   - Values: The videos themselves
2. **Self-attention visualization** showing how words relate to each other:

   Example: "The animal didn’t cross the street because it was too tired" where **"it"** strongly attends to **"animal"**
3. **Positional encoding heatmap:** Shows sine/cosine patterns that encode position information

Key Connections to Emphasize

1. **RNNs → Transformers:** Transformers solve RNN limitations by replacing sequential processing with parallel attention
2. **Self-attention → Multi-head attention:** Multi-head allows model to focus on different aspects of input simultaneously
3. **Transformer → BERT:** BERT is essentially a pre-trained transformer encoder with specific training objectives
4. **Attention → Classification:** How self-attention allows aggregation of information for classification tasks

Quiz Questions for Self-Testing

1. **Q:** What is the primary advantage of transformer architecture over RNNs?
   A: Parallel processing of input sequences, allowing for more efficient training and better capture of long-range dependencies.
2. **Q:** Why do we divide by $\sqrt{d_k}$ in the attention formula?
   A: To prevent the dot products from growing too large in magnitude, which would lead to extremely small gradients during backpropagation.
3. **Q:** Explain the difference between "permutation-equivariant" and "permutation-invariant".
   A: Permutation-equivariant means the output changes in a corresponding way when the input order changes. Permutation-invariant means the output remains the same regardless of input order. Self-attention is permutation-equivariant.
4. **Q:** Why is positional encoding necessary in transformers?
   A: Since transformers process all tokens in parallel with self-attention (which is permutation-equivariant), they need positional encoding to capture sequence order information.
5. **Q:** How does the [CLS] token approach differ from average pooling for classification?
   A: The [CLS] token learns to aggregate sequence information through self-attention, while average pooling simply takes the mean of all token representations.
6. **Q:** What are the components of a typical transformer encoder layer?
   A: Multi-head self-attention, Add & Norm (residual connection and layer normalization), Feed-forward network, and another Add & Norm.

Often Overlooked Details Examiners May Ask About

1. **Scaling factor in attention:** The $\sqrt{d_k}$ term is crucial for gradient stability and often overlooked.
2. **Feed-forward networks in transformers:** These are applied position-wise and typically expand dimension then contract:

$$\text{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2$$

3. **Masked self-attention in decoder:** Ensures predictions only depend on known outputs during training.
4. **Implementation of multi-head attention:** Involves splitting dimensions rather than separate passes through the attention mechanism.
5. **Residual connections:** Critical for training deep transformer networks by helping gradient flow.
6. **Layer normalization vs. batch normalization:** Transformers use layer norm which normalizes across features rather than batch examples.
7. **Computational complexity of self-attention:** $\mathcal{O}(n^2 d)$ where n is sequence length and d is embedding dimension – quadratic in sequence length.

Note: Examiners often ask you to derive or explain the attention mechanism in detail, so be prepared to write out the formula and explain each component's purpose!

2. Lecture 2: Visual Transformers

2.1 Foundational Concepts

Evolution from NLP to Vision

- **Original Transformer (2017):** *"Attention is All You Need"* paper – revolutionized NLP
- **Key insight:** "An image is worth 16×16 words" – treating image patches as tokens
- **Paradigm shift:** Moving from convolution-based architectures to attention-based ones
- **Main advantage:** Global receptive field from the start (vs. CNNs' local receptive fields)

Core Differences: NLP vs. Vision Transformers

Aspect	NLP Transformers	Vision Transformers
Input	1D sequence of word tokens	2D grid of pixels converted to patch tokens
Positional encoding	1D sequence positions	2D spatial positions (learned, not sinusoidal)
Scale challenges	Hundreds/thousands of tokens	High-resolution images → millions of pixels
Inductive bias	Minimal language structure bias	No built-in spatial locality (unlike CNNs)
Data requirements	Moderate	Higher (to compensate for lack of inductive bias)

2.2 Vision Transformer (ViT) Architecture

Input Processing Pipeline

1. **Patch extraction:** Divide image into non-overlapping patches (typically 16×16 pixels)
2. **Linear projection:** Flatten patches and project to embedding dimension (D)
3. **Class token:** Add learnable [CLS] token for classification (like BERT)
4. **Positional embedding:** Add learned positional embeddings to retain spatial information

Core Components

- **Transformer Encoder:** Series of Multi-Head Self-Attention layers and MLPs
- **Self-Attention:** Same mechanism as in NLP transformers

- **MLP Head:** Final classification layer applied to [CLS] token output

ViT Variants and Scaling

- **Size variants:** Base (L=12, H=768), Large (L=24, H=1024), Huge (L=32, H=1280)
- **Training efficiency:** ViT with 2.5k TPU days outperforms ResNet with 9.9k TPU days
- **Patch size impact:** Smaller patches → more tokens → higher computational cost but better performance

2.3 Hierarchical Vision Transformers

Swin Transformer

- **Key innovation:** Hierarchical structure with shifted windows of attention
- **Window-based attention:** Compute self-attention within local windows to reduce complexity
- **Shifting windows:** Alternate layers shift attention windows to enable cross-window connections
- **Progressive downsampling:** Merge neighboring patches to create hierarchical representation
- **Complexity:** Linear complexity with image size (vs. quadratic in vanilla ViT)

Advantages

- **Efficiency:** $\mathcal{O}(n)$ complexity vs. $\mathcal{O}(n^2)$ in vanilla transformers
- **Multi-scale processing:** Better suited for dense prediction tasks (detection, segmentation)
- **Performance:** Strong results with less computational resources

2.4 Object Detection with Transformers

DETR (DEtection TRansformer)

End-to-end approach: No need for hand-designed components like anchors and NMS

Architecture

1. CNN backbone extracts features
2. Transformer encoder processes these features
3. Transformer decoder with object queries predicts boxes directly
4. Feed-forward networks convert query outputs to box coordinates and class labels

Bipartite Matching Loss

- **Key innovation:** One-to-one assignment between predictions and ground truth
- **Contrast with traditional methods:** Traditional detectors use one-to-many assignment with NMS
- **Implementation:** Hungarian algorithm to find optimal matching
- **Training formulation:** Predefined number of queries (N) matched to ground truth boxes plus "no object" fillers

Benefits

- **Simplified pipeline:** Direct set prediction without post-processing
- **Global reasoning:** Objects predicted in relation to each other

- **Parallel decoding:** All objects predicted simultaneously

2.5 Image Segmentation with Transformers

Key Architectures

- **SETR:** Encoder-only approach with CNN decoder for upsampling
- **SegFormer:** Efficient hierarchical design with lightweight decoder
- **MaskFormer:** Query-based approach with mask predictions (treats segmentation as mask classification)
- **Mask2Former:** Improved MaskFormer with masked attention mechanism

Common Principles

- **Global context:** Transformers capture long-range dependencies important for segmentation
- **Query-based formulation:** Moving from per-pixel classification to mask prediction
- **Unified approach:** Same architecture works for semantic, instance, and panoptic segmentation

2.6 Advanced Concepts

Positional Encoding in Vision

- **Learned 1D embeddings:** Despite dealing with 2D images, ViT uses 1D position embeddings
- **Emergent properties:** Position embeddings learn to encode:
  - Distance (closer patches $\rightarrow$ similar embedding)
  - Row/column structure (patches in same row/col have similar embeddings)
  - 2D image topology (despite only being trained with 1D sequence)

Attention Visualization and Interpretability

- **Attention rollout:** Method to visualize attention flow through multiple layers
- **Last layer attention:** Shows which patches are most relevant for classification
- **Visual examples:** Shows transformers attend to semantically meaningful parts of images

Memory and Computational Challenges

- **Full self-attention:** $\mathcal{O}(n^2)$ complexity is problematic for high-resolution images
- **Strategies to address:**
  1. Window-based attention (Swin)
  2. Hierarchical architectures
  3. Linear attention approximations
  4. Hybrid CNN-Transformer architectures

2.7 Quiz Questions with Answers

1. **Q:** What is the fundamental difference in how ViT processes images compared to CNNs?
   A: ViT splits images into patches and processes them as a sequence using self-attention, whereas CNNs use local convolution operations with increasing receptive fields through the network.

2. **Q:** Why do Vision Transformers typically require more training data than CNNs?
   A: Vision Transformers lack the inductive biases of CNNs (locality, translation equivariance) and must learn these relationships from data, requiring more examples to generalize well.
3. **Q:** Explain the purpose of the [CLS] token in Vision Transformers.
   A: The [CLS] token is a learnable embedding prepended to the sequence of patch embeddings. Through self-attention, it aggregates information from all patches, and its final representation is used for image classification.
4. **Q:** How does the bipartite matching loss in DETR eliminate the need for NMS?
   A: Bipartite matching creates a one-to-one assignment between predictions and ground truth objects, forcing the model to output exactly one prediction per object. This inherently avoids duplicate detections, making NMS unnecessary.
5. **Q:** What is the key innovation of Swin Transformer that addresses the computational complexity of ViT?
   A: Swin Transformer computes self-attention within local windows rather than globally, reducing complexity from $\mathcal{O}(n^2)$ to $\mathcal{O}(n)$. It also uses shifted window patterns between layers to allow information flow across windows.
6. **Q:** How does MaskFormer change the approach to semantic segmentation compared to traditional methods?
   A: MaskFormer reformulates semantic segmentation from per-pixel classification to a set prediction problem, where the model predicts masks with associated class labels, similar to DETR's approach to object detection.

3. Lecture 3: Transformer Decoders, LLMs and GPTs

Transformer Architecture Overview

- **Transformer Decoder:** The component responsible for generating output sequences token by token
- **Essential Components:**
  1. Masked Multi-Head Attention (enforces autoregressive property)
  2. Multi-Head Cross-Attention (connects to encoder if present)
  3. Feed-Forward Networks
  4. Layer Normalization
  5. Residual connections

Key Transformer Model Types

- **Encoder-Decoder Models:** Translation, summarization (original Transformer)
- **Encoder-Only Models:** BERT, text classification, understanding
- **Decoder-Only Models:** GPT family, text generation, most modern LLMs

Attention Mechanisms

- **Self-Attention:** Token relationships within the same sequence
- **Masked Self-Attention:** Only allows a token to attend to previous tokens (slides 24–25)
- **Cross-Attention:** Allows decoder to attend to encoder outputs (slide 26)
- **Multi-Head Attention:** Allows multiple attention patterns in parallel

Autoregressive Generation (Critical for GPT-style models)

- Tokens generated sequentially, with each new token conditioned on all previous tokens
- **Formula:**

$$P(w_1, w_2, \ldots, w_n) = \prod_i P(w_i \mid w_1, w_2, \ldots, w_{i-1})$$

- Enforced via **causal masking** in the attention mechanism

3.1 GPT-Style Models vs BERT

GPT Architecture & Function

- **Decoder-only** transformer model
- **Unidirectional** (left-to-right) attention
- **Autoregressive** next token prediction
- Trained using standard language modeling objective
- Generate text by sampling from conditional probability distributions

BERT vs. GPT Comparison (Slide 32–33)

Feature	BERT	GPT
Model Type	Encoder-Only	Decoder-Only
Direction	Bidirectional	Unidirectional (left-to-right)
Pre-training	Masked Language Modeling (MLM)	Autoregressive Language Modeling
Fine-tuning	Task-specific layer added on top	Few-shot prompting or fine-tuning
Use Case	Understanding (classification, NER)	Generation (text completion, creative writing)
Original Creator	Google AI	OpenAI

Training Objectives

- **GPT:** Predict next token given previous tokens
- **BERT:** Predict masked tokens using bidirectional context

3.2 Machine Translation with Transformers

Encoder-Decoder for Translation

- **Input:** Source language sequence processed by encoder
- **Output:** Target language sequence generated by decoder
- **Cross-attention:** Decoder queries attend to encoder key-value pairs
- **Training objective:** Teacher forcing with target sequence

Translation Process (From Slides 12–23)

1. Source sentence tokenized and fed to encoder
2. Encoder builds contextual representations
3. Decoder starts with special start token
4. Decoder generates target tokens one-by-one
5. Cross-attention allows decoder to focus on relevant source tokens
6. Process continues until end token is generated

Difference from Decoder-Only Models

- Encoder-decoder has explicit source encoding step
- Cross-attention mechanism connects source and target
- More parameter-efficient for translation tasks
- Better handling of source language nuances

3.3 Key Diagrams & Workflows

Transformer Decoder Structure (Slide 24)

The presentation shows the decoder with:

- Masked Multi-Head Attention block
- Multi-Head Cross-Attention block (for encoder-decoder models)
- Feed-Forward Networks
- Layer normalization and residual connections

Masked Attention Mechanism (Slide 25)

- Shows how attention is masked to prevent looking at future tokens
- Contrast between standard self-attention and masked self-attention
- Implementation by setting future position scores to negative infinity

Autoregressive Decoder Workflow (Slides 17–22)

- Step-by-step illustration of how tokens are generated
- Starts with special token `<START>`
- Generates one token at a time
- Each new token depends on all previous tokens

Decoding Strategies (Slides 40–47)

- **Greedy decoding:** Always select highest probability token
- **Beam search:** Maintain multiple hypotheses
- **Sampling:** Draw from probability distribution (with temperature control)

3.4 Practice Quiz Questions

1. **Q:** What is the key difference between the attention mechanism in BERT and GPT?
   A: BERT uses bidirectional self-attention, while GPT uses masked (causal) self-attention that only allows a token to attend to previous tokens.
2. **Q:** Why can't GPT process all output tokens in parallel during inference?
   A: Due to the autoregressive nature, each token depends on previously generated tokens, forcing sequential generation.
3. **Q:** What is the purpose of cross-attention in an encoder-decoder transformer?
   A: Cross-attention allows the decoder to attend to relevant parts of the encoder's output, connecting source and target sequences.
4. **Q:** How does beam search differ from greedy decoding?
   A: Greedy decoding selects the most probable token at each step, while beam search maintains multiple hypotheses and selects the overall most probable sequence.
5. **Q:** Why might multinomial sampling with temperature control produce more diverse text than beam search?
   A: Sampling introduces randomness that helps avoid repetitive patterns, while temperature control allows balancing between diversity and coherence.
6. **Q:** What problem in tokenization does Byte Pair Encoding (BPE) address?
   A: BPE balances vocabulary size and sequence length by using subword units, handling rare

words better than word-level tokenization while being more efficient than character-level tokenization.

3.5 Common Pitfalls & Advanced Questions

Causal Masking in GPT

- **Mechanism:** Sets attention scores for future positions to negative infinity
- **Purpose:** Enforces autoregressive constraint
- **Pitfall:** Confusing with padding masks (different purpose)
- **Examiner might ask:** How does causal masking contribute to the training efficiency of GPT models?

Parallel Processing Limitations

- **Training:** Can be parallelized because ground truth is available
- **Inference:** Must be sequential due to autoregressive nature
- **Pitfall:** Assuming inference can be parallelized like training
- **Examiner might ask:** What approaches might improve inference speed in autoregressive models?

Long-Range Dependencies

- **Challenge:** Attention complexity grows quadratically with sequence length
- **Solutions:** Sparse attention, sliding windows, hierarchical attention
- **Pitfall:** Not understanding the computational bottlenecks
- **Examiner might ask:** How do modern LLMs handle context windows of 100K+ tokens?

Tokenization Tradeoffs

- **Character-level:** Small vocabulary, long sequences
- **Word-level:** Large vocabulary, OOV problems
- **Subword (BPE):** Compromise, $\sim$4 characters per token in English
- **Examiner might ask:** How might tokenization biases affect model performance across languages?

Decoding Strategy Selection

- **When to use beam search:** When the most probable overall sequence is desired
- **When to use sampling:** When diversity is important
- **Pitfall:** Using the wrong strategy for the application
- **Examiner might ask:** Why does beam search tend to produce repetitive text in open-ended generation?

Evaluation Challenges

- **Automated metrics:** BLEU, ROUGE have limitations
- **Human evaluation:** Subjective but often necessary
- **Pitfall:** Over-reliance on automated metrics
- **Examiner might ask:** How would you evaluate an LLM for factual accuracy versus creativity?

4. Lecture notes: Diffusion Models

1. Fundamental Concepts of Diffusion Models

Diffusion models belong to the class of generative models — models that describe a probability distribution p and can generate samples from that distribution. They joined other generative approaches like Variational Autoencoders (VAEs), normalizing flows, and Generative Adversarial Networks (GANs).

Denoising Diffusion Probabilistic Models (DDPMs) were introduced in the paper by Ho et al. [4]. Until their arrival, GANs were the dominant approach for high-quality image generation, despite being less probabilistically rigorous. DDPMs have since surpassed GANs in visual quality.

Key Differentiators from Other Generative Models

In traditional generative models like VAEs and GANs, we learn to predict data (images) from latent vectors that follow a predefined distribution (often Gaussian). Diffusion models encode images differently — by incrementally adding noise to the image over a pre-determined number (T) of time steps.

Two crucial differences from VAEs/GANs:

1. **No dimension reduction** — There is no inherent dimension reduction in the latent representations.
2. **No semantic structure** — The Step 0 latent representation contains no semantic or geometric information. Interpolating between two noised images will not generate semantically meaningful transformations.

4.1 The Forward Process (Diffusion/Noise Addition)

The forward process is formalized as a Markov process q that progressively adds noise to an image:

Definition 4.1.1 — Forward Process of Diffusion Model.

$$q(x_{1:T} \mid x_0) := \prod_{t=1}^{T} q(x_t \mid x_{t-1})$$

where

$$q(x_t \mid x_{t-1}) := \mathcal{N}(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t \mathbf{I})$$

Here, $\mathbf{I}$ is the identity covariance matrix over image space, and β_t values are hyperparameters referred to as the variance schedule.

Closed-form Sampling

A key insight is that we can derive a closed-form expression to directly sample a noisy image at any time step without having to sequentially add noise:

Definition 4.1.2 — Closed form sampling.

$$q(x_t \mid x_0) = \mathcal{N}(x_t; \sqrt{\bar{\alpha}_t}x_0, (1-\bar{\alpha}_t)\mathbf{I})$$

where $\alpha_t := 1 - \beta_t$ and $\bar{\alpha}_t := \prod_{s=1}^{t} \alpha_s$.

Variance Scheduling

For implementation, the original DDPM paper uses a linear variance schedule with β_t increasing from $\beta_1 = 10^{-4}$ to $\beta_T = 0.02$. Alternative schedules like the cosine schedule have also been proposed, which emphasize later denoising steps.

4.2 The Reverse Process (Denoising)

The reverse process is where the main work happens — the DDPM learns to generate noise-free images from their noisy representations. This iterative denoising is performed by a deep neural network that represents a joint distribution $p_\theta(x_{0:T})$.

Definition 4.2.1 — Reverse Process (Denoising). Like the forward process, the reverse process is a Markov chain:

$$p_\theta(x_{0:T}) := p(x_T) \prod_{t=1}^{T} p_\theta(x_{t-1} \mid x_t)$$

where

$$p_\theta(x_{t-1} \mid x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t))$$

The process typically starts with an initial noise image x_T drawn from a standard normal distribution:

$$p(x_T) = \mathcal{N}(x_T; 0, \mathbf{I})$$

4.3 Learning to Denoise

Rather than predicting the denoised image x_{t-1} directly from x_t, empirical research has found it easier to predict the noise ε that was used to generate x_t from the original noise-free image x_0.

Reparameterized Forward Equation

This uses the formula:

Definition 4.3.1 — Reparametrization Trick.

$$x_t = \sqrt{\bar{\alpha}_t} x_0 + \sqrt{1-\bar{\alpha}_t}\,\varepsilon$$

The model is trained to predict $\varepsilon_\theta(x_t, t)$, which estimates the noise ε from x_t.

Architecture

- U-Net architecture is widely used as it's effective for image-to-image tasks
- Time point t is included as an additional input
- At every down- and upsampling step, time is incorporated as a broadcast channel
- This broadcasted time representation is added to the image, similar to positional encoding

Loss Function

The DDPM paper derives a Bayesian loss, but in practice a simpler MSE loss is used:

Definition 4.3.2 — Loss Function.

$$L_{\text{simple}}(\theta) := \mathbb{E}_{x_0,\varepsilon,t}\left[\left\|\varepsilon - \varepsilon_\theta\left(\sqrt{\bar{\alpha}_t} x_0 + \sqrt{1-\bar{\alpha}_t}\,\varepsilon, t\right)\right\|^2\right]$$

This is simply the mean squared error (MSE) between the predicted noise and the true noise ε.

4.4 Training and Sampling Process

Training

At each training iteration:

1. Sample a random x_0 from data
2. Sample a random time point t uniformly from the T time points
3. Sample random noise $\varepsilon \sim \mathcal{N}(0, \mathbf{I})$
4. Compute the noisy image:

   Definition 4.4.1 — Noisy image at time t.

   $$x_t = \sqrt{\bar{\alpha}_t} x_0 + \sqrt{1-\bar{\alpha}_t}\,\varepsilon$$

5. Make a gradient update of the MSE loss w.r.t. θ

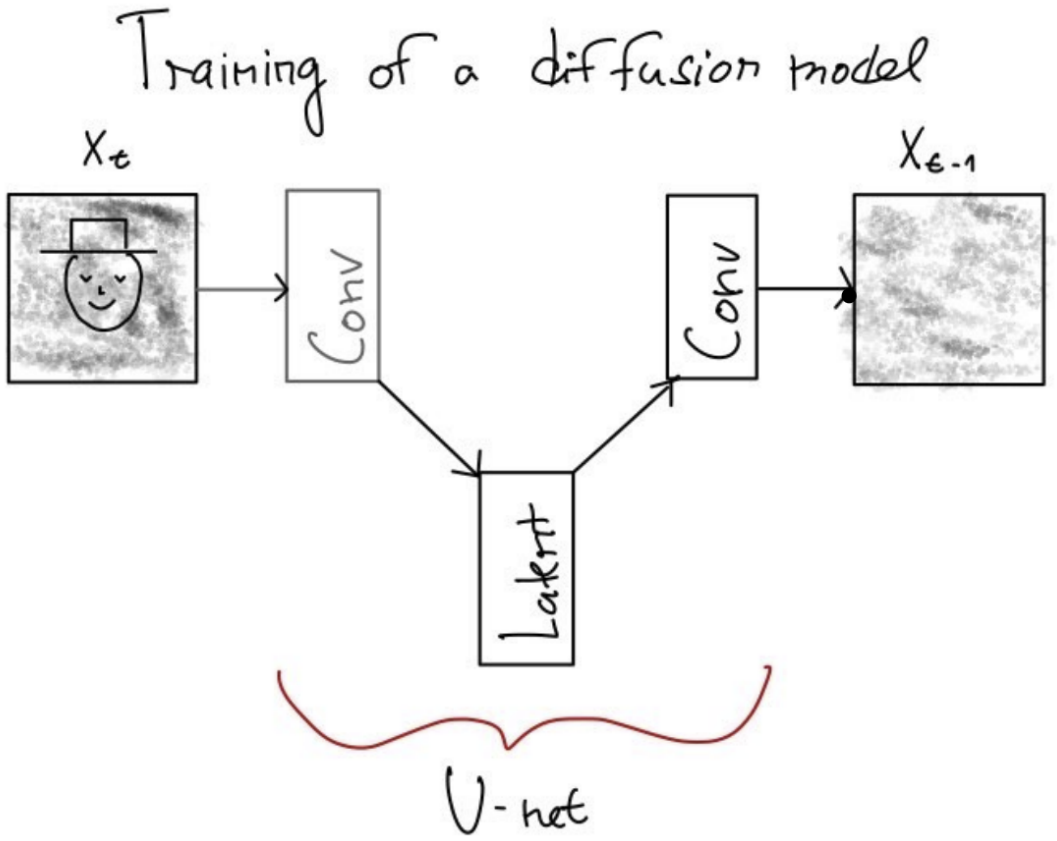


Figure 4.1: Illustration of the denoising diffusion process using a U-Net architecture. At each denoising step, the U-Net model ε_θ predicts the noise component ε present in the noisy image x_t. Rather than directly predicting the clean image, the model outputs the estimated noise $\hat{\varepsilon}_\theta(x_t,t)$, which is then subtracted from x_t (with appropriate scaling based on the diffusion schedule) to obtain the less noisy image approximation x_{t-1}. This approach, where the network is trained to minimize $\mathbb{E}[\|\varepsilon - \hat{\varepsilon}_\theta(x_t,t)\|^2]$, has been shown to provide more stable training dynamics than directly predicting the denoised image.

Sampling

For sampling from the model:

1. Set covariance $\Sigma_\theta(x_t,t) = \sigma_t^2\mathbf{I}$, where $\sigma_t^2 = \beta_t$ (simplest version)
2. Set mean:

Definition 4.4.2 — Denoising Mean Equation:.

$$\mu_\theta(x_t,t) = \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\varepsilon_\theta(x_t,t)\right)$$

3. Sample $x_{t-1} \sim \mathcal{N}(\mu_\theta, \Sigma_\theta)$
4. Repeat iteratively to sample x_{t-1} from x_T

4.5 Key Exam Topics and Common Questions

What makes diffusion models different from VAEs and GANs?

- No dimension reduction in latent space
- Latent variables have no semantic structure
- Progressive noise addition/removal rather than direct encoding/decoding
- More stable training compared to GANs
- Better image quality than earlier models

What is the variance schedule $\beta_1 \dots \beta_T$ and why is it important?

- Controls how quickly noise is added in the forward process
- Affects the difficulty of the denoising task
- Different schedules (linear, cosine) have different properties
- Cosine schedule focuses learning more on later denoising steps

Why do we predict noise ε rather than directly predicting x_{t-1}?

- Empirically proven to be easier and more effective
- Allows for a simpler loss function (MSE)
- Results in better performance

What architecture is typically used for diffusion models?

- U-Net architecture (originally designed for image segmentation)
- Modified to incorporate time as an additional input
- Time is broadcast as a channel and added at each scale

How is the sampling process defined in diffusion models?

- Start with pure noise $x_T \sim \mathcal{N}(0, \mathbf{I})$
- Iteratively denoise step by step using the learned model
- Each step involves sampling from a Gaussian with learned parameters
- Final result x_0 is the generated sample

4.6 Practice Questions for Exam Preparation

1. **Q:** Explain the mathematical formulation of the forward process in diffusion models.
   A: The forward process is a Markov chain that progressively adds Gaussian noise to an image:

$$q(x_{1:T} \mid x_0) := \prod_{t=1}^{T} q(x_t \mid x_{t-1}) \quad \text{where} \quad q(x_t \mid x_{t-1}) := \mathcal{N}(x_t; \sqrt{1-\beta_t} x_{t-1}, \beta_t \mathbf{I})$$

   This can be reparameterized to directly sample x_t from x_0:

$$q(x_t \mid x_0) = \mathcal{N}(x_t; \sqrt{\bar{\alpha}_t} x_0, (1-\bar{\alpha}_t)\mathbf{I})$$

2. **Q:** What is being learned by the neural network in a DDPM?
   A: The neural network is learning to predict the noise ε that was added to the original image, given a noisy image x_t and time step t. This is then used to construct the parameters (mean and variance) of the reverse process distributions.
3. **Q:** Why is the U-Net architecture suitable for diffusion models?
   A: The U-Net is designed for image-to-image tasks where spatial relationships matter. It has skip connections that help preserve spatial information. In diffusion models, we're essentially doing image-to-image tasks: predicting noisy or denoised images with the same dimensions as the input.
4. **Q:** How does the training objective differ from the theoretical formulation in the DDPM paper?
   A: While the theoretical loss is based on a variational lower bound, in practice a simpler MSE loss is used between the predicted noise and the actual noise. This was found to work better empirically.
5. **Q:** What extensions to diffusion models might be covered in the next lecture?
   A: The next lecture will cover conditioning and guidance techniques, including classifier guidance, classifier-free guidance, universal guidance, and counterfactual explanations via diffusion models.

4.7 Common Pitfalls and Deep Insights

- **Latent Space Structure:** Unlike VAEs, the latent space in diffusion models lacks semantic structure, so traditional interpolation doesn't work for tasks like "making a person smile."
- **Computational Cost:** Diffusion models require sequential sampling through T steps, making them slower than single-pass generators like GANs.
- **Theoretical vs. Practical:** There's often a gap between theory and what works best in practice (e.g., simplified MSE loss performs better than variational losses).
- **Hyperparameter Sensitivity:** The variance schedule significantly affects performance and requires careful tuning.
- **Connection to Score-Based Models:** Diffusion models are closely related to score-based generative models – this insight may come up in deeper theoretical discussions.

Reflection Prompt: Be prepared to explain how you would approach the task of *"making a person smile"* with a diffusion model, as this was posed as a discussion point in the notes.

5. Lecture 4: Diffusion Models

5.1 Overview of Diffusion Models

Diffusion models generate data by reversing a gradual noising process. Unlike other generative models that map from a fixed latent distribution to data, diffusion models:

- Encode images by progressively adding noise over T time steps
- Learn to reverse this process by denoising step-by-step
- Use noisy images as time-dependent latent variables

Key Distinction from Other Generative Models

Traditional generative models (GANs, VAEs) map latent vectors to data directly. In contrast, diffusion models:

- Work with a sequence of increasingly noisy versions of the data
- Define both forward (noising) and reverse (denoising) Markov processes
- Train a neural network to predict and remove noise at each step

5.2 The Forward and Reverse Processes

Forward Process (Adding Noise)

The forward process is a Markov chain that gradually adds Gaussian noise:

- **Mathematical formulation:**

$$q(x_{1:T} \mid x_0) := \prod_{t=1}^{T} q(x_t \mid x_{t-1}), \quad q(x_t \mid x_{t-1}) := \mathcal{N}(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t \mathbf{I})$$

- **Variance schedule:** $\beta_1, \beta_2, \ldots, \beta_T$ determines noise scaling
  - Ho et al. use a linear schedule from $\beta_1 = 10^{-4}$ to $\beta_T = 0.02$
  - Nichol et al. later improved this with a cosine schedule
- **Closed-form sampling:**

$$q(x_t \mid x_0) = \mathcal{N}(x_t; \sqrt{\bar{\alpha}_t}x_0, (1-\bar{\alpha}_t)\mathbf{I})$$

where $\alpha_t := 1 - \beta_t$ and $\bar{\alpha}_t := \prod_{s=1}^{t} \alpha_s$

- **Reparameterization trick:**

$$x_t(x_0, \varepsilon) = \sqrt{\bar{\alpha}_t} x_0 + \sqrt{1 - \bar{\alpha}_t}\, \varepsilon, \quad \varepsilon \sim \mathcal{N}(0, \mathbf{I})$$

Reverse Process (Generative Process)

The reverse process is also a Markov chain, but with learned Gaussian transitions:

- **Mathematical formulation:**

$$p_\theta(x_{0:T}) := p(x_T) \prod_{t=1}^{T} p_\theta(x_{t-1} \mid x_t), \quad p_\theta(x_{t-1} \mid x_t) := \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t))$$

  Initialized with $p(x_T) = \mathcal{N}(x_T; 0, \mathbf{I})$
- **Key insight:** Instead of predicting x_{t-1} directly, we predict the noise ε that was added
  - The model $\varepsilon_\theta(x_t, t)$ is trained to estimate ε from the noisy image x_t
- **Sampling equation:**

$$x_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left(x_t - \frac{1 - \alpha_t}{\sqrt{1 - \bar{\alpha}_t}} \varepsilon_\theta(x_t, t) \right) + \sigma_t z$$

  Where $\sigma_t = \beta_t$ and z is random noise ($z = 0$ at final step)

5.3 Model Architecture and Training

Architecture

- **U-Net backbone:** The noise predictor ε_θ uses a U-Net architecture.
- **Time conditioning:** The time step t is an additional input, broadcast and added to the image representation at each layer (similar to positional encoding).

Training Objective

- **Theoretical foundation:** While a variational bound (ELBO) exists, a simpler loss works better in practice.
- **Simplified loss:**

$$L_{\text{simple}}(\theta) := \mathbb{E}_{x_0, \varepsilon, t} \left[\left\| \varepsilon - \varepsilon_\theta \left(\sqrt{\bar{\alpha}_t} x_0 + \sqrt{1 - \bar{\alpha}_t}\, \varepsilon, t \right) \right\|^2 \right]$$

- **Intuition:** The model directly predicts the noise that was added to create x_t from x_0.

Training Algorithm

1. Sample x_0 from the data distribution
2. Sample a random time step $t \sim \text{Uniform}(\{1, \ldots, T\})$
3. Sample random noise $\varepsilon \sim \mathcal{N}(0, \mathbf{I})$
4. Create noisy sample:

$$x_t = \sqrt{\bar{\alpha}_t} x_0 + \sqrt{1 - \bar{\alpha}_t}\, \varepsilon$$

5. Train the model to predict ε given x_t and t

Sampling/Generation Algorithm

1. Sample $x_T \sim \mathcal{N}(0, \mathbf{I})$ (pure noise)
2. For $t = T, T-1, \ldots, 1$:
   - Sample $z \sim \mathcal{N}(0, \mathbf{I})$ if $t > 1$, else $z = 0$
   - Compute x_{t-1} using the model prediction
3. Return x_0 (the generated image)

5.4 Important Visual Examples

The slides illustrate the diffusion process with several key visualizations:

- **Forward (noising) process:** A sprite image gradually becomes more noisy across time steps
- **Reverse (denoising) process:** Pure noise gradually forms into a coherent sprite image
- **Generated samples:** Examples of faces generated by the model

These visualizations help understand how:

- Initial data structure remains visible even with significant noise
- The reverse process reconstructs the data distribution step-by-step
- Different noise schedules affect the quality of latent representations

5.5 Important Visual Examples

The slides illustrate the diffusion process with several key visualizations:

- **Forward (noising) process:** A sprite image gradually becomes more noisy across time steps
- **Reverse (denoising) process:** Pure noise gradually forms into a coherent sprite image
- **Generated samples:** Examples of faces generated by the model

These visualizations help understand how:

- Initial data structure remains visible even with significant noise
- The reverse process reconstructs the data distribution step-by-step
- Different noise schedules affect the quality of latent representations

5.6 Quiz Questions and Answers

Q1: What is the main difference between diffusion models and other generative models like GANs or VAEs?
A1: In diffusion models, the latent space consists of noisy versions of the data across multiple time steps, rather than a single lower-dimensional latent space. Diffusion models gradually add noise in the forward process and learn to reverse this process, while GANs and VAEs learn direct mappings between latent space and data space.

Q2: What does the denoising model $\varepsilon_\theta(x_t, t)$ actually learn to predict?
A2: The model learns to predict the noise ε that was added to the original image x_0 to create the noisy image x_t, rather than directly predicting the clean image.

Q3: Why is the variance schedule $\beta_1, \ldots, \beta_T$ important in diffusion models?
A3: The variance schedule controls how quickly noise is added in the forward process. It affects the trade-off between sample quality and generation speed. A well-designed schedule (like the cosine schedule) ensures noise is added at an appropriate rate across all time steps.

Q4: How is the time step t incorporated into the model architecture?
A4: The time step t is broadcast and added to the image representation at every up/downsampling step of the U-Net, similar to positional encoding in transformers.

Q5: What is the key insight that allows for a simpler loss function in DDPMs?
A5: The key insight is that instead of optimizing the complex variational bound directly, the model can be trained to simply predict the noise that was added during the forward process, using a mean squared error loss between predicted and actual noise.

Q6: How does sampling differ from training in diffusion models?
A6: In training, we start with real data x_0, add noise to create x_t, and train the model to predict the added noise. In sampling, we start with pure noise x_T and iteratively apply the reverse process $p_\theta(x_{t-1} \mid x_t)$ to gradually denoise until we reach x_0.

5.7 Details Often Overlooked

Theoretical Connection to Score-Based Models

- The noise prediction objective is equivalent to score matching
- $\varepsilon_\theta(x_t, t)$ approximates the score function $\nabla_x \log p(x)$
- This connects DDPMs to score-based generative models and Langevin dynamics

Simplified Covariance Choice

- The reverse process covariance $\Sigma_\theta(x_t, t)$ is fixed as $\sigma_t^2 \mathbf{I}$ with $\sigma_t^2 = \beta_t$
- This simplification works well in practice but is a modeling choice that could be learned

Impact of Variance Schedule on Sampling

- **Linear schedule:** Later timesteps are almost pure noise
- **Cosine schedule:** Adds noise more gradually throughout process
- The schedule choice affects both training stability and generation quality

Reuse of Noise Prediction for Different Tasks

The same basic DDPM framework can be extended to:

- Conditional generation (guidance)
- Image-to-image translation
- Inpainting and manipulation

Efficiency Considerations

- The iterative nature of diffusion models makes them slower than GANs/VAEs for generation
- Various approaches exist to reduce the number of sampling steps required

5.8 Additional Study Tips

1. **Understand the equations:** Work through the derivation of the sampling equation to understand why it works.
2. **Practice tracing through the algorithms:** Be able to describe both the training and sampling algorithms step by step.

3. **Compare with other models:** Be prepared to compare diffusion models with GANs, VAEs, and normalizing flows.
4. **Consider limitations:** Discuss the computational cost of the iterative sampling process and methods to address it.
5. **Follow-up developments:** Familiarize yourself with classifier-free guidance (Ho and Salimans) and improved noise schedules (Nichol et al.) mentioned in the slides.

6. Lecture 5: Diffusion Guidance

6.1 Guidance in Diffusion Models: Comprehensive Exam Preparation

1. Introduction to Guidance in Diffusion Models

Guidance is a fundamental technique in diffusion models that combines generative image models with additional input information to control the generation process. As the slides indicate, the core concept is to steer the diffusion sampling process toward generating images with specific desired properties.

Key Insight: Guidance modifies the sampling trajectory of a diffusion model by introducing additional terms that *"pull"* the generation process toward specific outcomes.

6.2 Types of Guidance

6.2.1 2.1 Classifier Guidance

Classifier guidance uses a pretrained classifier to steer diffusion models toward generating images of specific classes.

Algorithm and Mechanism

- Start with a trained, unconditioned diffusion model.
- At each sampling step t (going from T to 1):
  - Compute mean (μ) and variance (Σ) from the diffusion model.
  - Apply the classifier gradient:

$$x_{t-1} \leftarrow \text{sample from } \mathcal{N}(\mu + s\Sigma\nabla_{x_t} \log p_\phi(y \mid x_t), \Sigma)$$

  - Here, $\nabla_{x_t} \log p_\phi(y \mid x_t)$ is the gradient pointing in the direction of the desired class.
  - s is a scale factor that amplifies guidance strength.

Key Equation:

$$x_{t-1} \leftarrow \text{sample from } \mathcal{N}(\mu + s\Sigma\nabla_{x_t} \log p_\phi(y \mid x_t), \Sigma)$$

Effect of Guidance Scale s

- Higher s produces more class-consistent images, but reduces diversity.
- Example from slides: Pembroke Welsh Corgi
  - $s = 1.0$: FID = 33.0
  - $s = 10.0$: FID = 12.0
- **Mathematical Insight:**

$$s \cdot \nabla_{x_t} \log p(y \mid x) = \nabla_{x_t} \log \left(\frac{1}{Z} \cdot p(y \mid x)^s \right)$$

- Increasing s sharpens the distribution

6.2.2 2.2 Classifier-Free Guidance

Classifier-free guidance eliminates the need for a separate classifier by jointly training conditional and unconditional diffusion models.

Key Mechanism

- During training:
  - Randomly discard conditioning with probability p_{uncond} to train both conditional and unconditional models jointly.
  - Use one-hot encoding for class conditioning.
- During sampling:
  - Combine predictions from both conditional and unconditional models with a weighting factor.

Advantages over Classifier Guidance

- No need to train a separate classifier on noisy data
- Simpler training pipeline
- Avoids potential adversarial attack scenarios

6.3 Universal Guidance

Universal guidance generalizes the concept of guidance to arbitrary functions and loss terms.

Key Insight: If $f(x)$ is any prediction function on image x, and $\mathscr{L}$ is a loss function, then guidance by minimizing $\mathscr{L}(c, f(x))$ optimizes for images x where $f(x) \approx c$.

Applications shown in slides

1. **Segmentation guidance:** Guide generation using cross-entropy loss between desired segmentation mask and predicted segmentation of generated image.
2. **Object location guidance:** Using object detection loss with bounding boxes.
3. **Style guidance:** Using cosine similarity between CLIP embeddings of style image and generated image.

6.4 Counterfactual Explanations with Diffusion Guidance

Counterfactual explanations answer the question: *"How should the input change to obtain a different classification?"*

Implementation with Diffusion Models

- The diffusion model provides the data compatibility term (keeps counterfactual realistic).
- A form of classifier guidance provides the direction toward the target class.
- **Challenge:** Need for noise-aware classifiers.
- **Solution approaches:**
  1. Iterative denoising with gradient steps at each time step (expensive, $\mathcal{O}(T^2)$ operations).
  2. Use closed-form approximation of noise-free image to compute gradient more efficiently.

6.5 Key Implementation Details and Challenges

5.1 Noise-Level Aware Classifiers

For effective classifier guidance, the classifier must be trained on noisy images at various diffusion timesteps. This is challenging when using off-the-shelf black-box classifiers.

5.2 Efficiency in Guidance Computation

- **Naïve approach:** Requires $\mathcal{O}(T^2)$ denoising steps — very expensive.
- **Optimized approach:** Use closed-form approximation to estimate clean image from noisy image:

$$x_0 = \frac{x_t(x_0, \varepsilon) - \sqrt{1-\bar{\alpha}_t}\,\varepsilon}{\sqrt{\bar{\alpha}_t}}$$

- This allows computing the classifier gradient directly without completing full denoising.

6.6 Evaluation Metrics for Guided Diffusion

6.1 Fréchet Inception Distance (FID)

- Measures similarity between generated and real images
- Computes 2-Wasserstein distance between feature distributions extracted by InceptionNet
- Lower FID indicates better quality and higher similarity to training data

6.2 Inception Score

- Originates from GAN literature
- Uses pretrained InceptionNet to evaluate sample quality
- Combines:
  - Low entropy of conditional label distribution (sharp, classifiable images)
  - High entropy of marginal distribution (diversity across classes)

6.3 Precision and Recall

- **Precision:** Measures quality of generated samples
- **Recall:** Measures coverage of the data distribution

6.7 Oral Exam-Style Quiz Questions & Answers

Q1: What is the key difference between classifier guidance and classifier-free guidance?
A1: Classifier guidance uses a separately trained classifier to guide the diffusion process, requiring

the classifier to understand noisy images at various timesteps. In contrast, classifier-free guidance jointly trains conditional and unconditional diffusion models, eliminating the need for a separate classifier. This simplifies the pipeline and avoids the challenge of training classifiers on noisy data.

Q2: How does the guidance scale parameter 's' affect generated images?
A2: The guidance scale s controls the strength of the guidance signal. Higher values of s make the model follow the guidance more strongly, resulting in images that better match the desired condition (higher fidelity) but reduce diversity. Mathematically, it sharpens the classifier's distribution by raising probabilities to the power of s.

Q3: Why are one-hot encodings important for class conditioning in classifier-free guidance?
A3: One-hot encodings provide clear, distinct representations for each class, helping the model learn separate, well-defined distributions. This is essential for effective class conditioning and for learning meaningful inter-class differences.

Q4: How does universal guidance extend the concept of classifier guidance?
A4: Universal guidance generalizes classifier guidance to any differentiable function $f(x)$ and loss function $\mathscr{L}$. Instead of using only a classifier's gradient, it can use gradients from segmentation, detection, or style models to optimize arbitrary criteria $\mathscr{L}(c, f(x))$, where c is the desired target and $f(x)$ is the model output on image x.

Q5: What is the main challenge in using diffusion models for counterfactual explanations?
A5: Classifier guidance usually needs classifiers trained on noisy images, but counterfactual use cases often involve off-the-shelf classifiers trained only on clean images. This mismatch requires methods to bridge the gap between classifiers and the noisy states in the diffusion process.

6.8 Common Misconceptions and Subtleties

Misconception #1: Classifier guidance simply adds a gradient term to the sampling process
Reality: It's more nuanced – the gradient is scaled by the covariance matrix Σ of the diffusion model at that timestep, ensuring guidance is appropriately calibrated to the noise level.

Misconception #2: Higher guidance scales always lead to better results
Reality: There's a trade-off. Higher guidance scales increase fidelity to the condition but reduce diversity, and can introduce artifacts or unnatural images if pushed too far.

Misconception #3: Classifier-free guidance is always superior to classifier guidance
Reality: Each method has advantages. Classifier guidance can leverage strong pretrained classifiers, while classifier-free guidance simplifies training but requires more compute. The best choice depends on available tools and objectives.

Misconception #4: Guidance is applied uniformly throughout the diffusion process
Reality: The effect of guidance varies over time. Early timesteps (high noise) benefit from stronger guidance to shape structure, while later timesteps (low noise) require less aggressive guidance to preserve detail.

6.9 Advanced Applications and Recent Developments

The slides highlight emerging use cases beyond basic class conditioning:

- Using guidance for counterfactual explanations in image classifiers
- Universal guidance for segmentation, object placement, and style transfer
- Medical applications using counterfactual examples (e.g., detecting surgical markers or tubes)

These demonstrate how diffusion guidance evolved from class-based conditioning to powerful tools for controlled generation across many domains and tasks.

6.10 Common Misconceptions and Subtleties

Misconception #1: Classifier guidance simply adds a gradient term to the sampling process
Reality: It's more nuanced – the gradient is scaled by the covariance matrix Σ of the diffusion model at that timestep, ensuring guidance is appropriately calibrated to the noise level.

Misconception #2: Higher guidance scales always lead to better results
Reality: There's a trade-off. Higher guidance scales increase fidelity to the condition but reduce diversity, and can introduce artifacts or unnatural images if pushed too far.

Misconception #3: Classifier-free guidance is always superior to classifier guidance
Reality: Each method has advantages. Classifier guidance can leverage strong pretrained classifiers, while classifier-free guidance simplifies training but requires more compute. The best choice depends on available tools and objectives.

Misconception #4: Guidance is applied uniformly throughout the diffusion process
Reality: The effect of guidance varies over time. Early timesteps (high noise) benefit from stronger guidance to shape structure, while later timesteps (low noise) require less aggressive guidance to preserve detail.

6.11 Advanced Applications and Recent Developments

The slides highlight emerging use cases beyond basic class conditioning:

- Using guidance for counterfactual explanations in image classifiers
- Universal guidance for segmentation, object placement, and style transfer
- Medical applications using counterfactual examples (e.g., detecting surgical markers or tubes)

These demonstrate how diffusion guidance evolved from class-based conditioning to powerful tools for controlled generation across many domains and tasks.

7. Lecture 6: SOTA Text2Image Generation

7.1 Core Concepts of Latent Diffusion Models (LDMs)

Problem with Traditional Diffusion Models

Traditional Denoising Diffusion Probabilistic Models (DDPMs) face significant limitations:

- Limited to $\sim 256 \times 256$ resolution images
- Computationally expensive since they operate directly in pixel space
- As noted in the slides:

> "Most bits of a digital image correspond to imperceptible details... gradients and neural network backbone need to be evaluated on all pixels, leading to superfluous computations."

Latent Diffusion Solution

LDMs solve this by:

1. **Compressing images into latent representations** using an autoencoder (VAE)
2. **Performing diffusion in this compressed latent space**
3. **Decoding back to pixel space** after the diffusion process

Advantages:

- **Efficiency:** Diffusion happens in a much smaller dimensional space
- **Focus on semantics:** The latent space primarily captures meaningful image content

LDM Architecture Components

1. **Encoder (E):** Compresses image x into latent representation z
2. **Latent Space:** Where diffusion occurs (forward noising process and reverse denoising)
3. **UNet-based Denoising Network:** Predicts noise at each denoising step
4. **Decoder (D):** Transforms the final latent representation back to pixel space
5. **Conditioning Mechanism:** Enables control through text or other modalities

7.2 Detailed Look at Latent Diffusion Model Architecture

Autoencoder (VAE) Component

The autoencoder serves two critical functions:

- **Encoder:** Compresses high-dimensional pixel space into compact latent representation
- **Decoder:** Reconstructs pixel information from latent representation after diffusion

This compression focuses on perceptually relevant features rather than pixel-perfect reconstruction, as visualized in the slides through rate-distortion curves showing the trade-off between compression and quality.

Diffusion in Latent Space

The diffusion process follows these steps:

1. **Forward Process:** Gradually add Gaussian noise to latent representation

$$z_0 \to z_1 \to z_2 \to \cdots \to z_T \quad \text{(pure noise)}$$

2. **Reverse Process (Denoising):**
   - Start with random noise z_T
   - Iteratively predict and remove noise:

$$z_T \to z_{T-1} \to \cdots \to z_1 \to z_0$$

   - Use U-Net architecture with cross-attention mechanisms for denoising
3. **Key Advantage:** Operating in latent space means the diffusion model handles significantly fewer dimensions:
   - If image is $512 \times 512 \times 3$ pixels $\approx$ 786K dimensions
   - Latent representation might be $64 \times 64 \times 4 \approx$ 16K dimensions (approx. $50\times$ reduction)

7.3 CLIP: The Vision-Language Model Powering Text Conditioning

CLIP Architecture and Training

CLIP consists of:

1. **Image Encoder:** Processes images into embeddings
2. **Text Encoder:** Processes text prompts into embeddings
3. **Shared Embedding Space:** Where both modalities are aligned

CLIP is trained using a contrastive loss approach:

- **Goal:** Maximize similarity between paired image-text samples and minimize it for unpaired ones
- **Training Data:** Large-scale image-text pairs from the internet
- **Loss Function:** Each image predicts which caption matches from a batch of options

How CLIP Enables Zero-Shot Classification

As shown in slide p.37, CLIP allows zero-shot classification by:

1. Creating text templates: `"A photo of a {class}"`
2. Computing text embeddings for all class names
3. Computing image embedding for the target image
4. Finding the text embedding with the highest similarity to the image

CLIP for Text-to-Image Generation

CLIP's role in text-to-image systems:

1. Encodes text prompts into embeddings that guide the diffusion process
2. Provides a way to measure similarity between generated images and text descriptions
3. Enables classifier-free guidance by comparing conditional and unconditional generation

7.4 Text-to-Image Generation End-to-End (Stable Diffusion)

Stable Diffusion Pipeline

The complete text-to-image generation process:

1. **Text Encoding:**
   - Text prompt is tokenized and processed by CLIP's text encoder
   - Produces text embeddings representing the desired image content
2. **Latent Diffusion Process:**
   - Start with random noise in latent space
   - Apply UNet-based denoising model conditioned on text embeddings
   - Cross-attention layers inject text information into the spatial features
   - Iteratively denoise until a clean latent representation is produced
3. **Image Decoding:**
   - VAE decoder transforms the clean latent representation to pixel space
   - Produces the final generated image matching the text description

Cross-Attention Mechanism

Cross-attention is the critical component for conditioning:

- **Query:** Features from the diffusion U-Net
- **Key/Value pairs:** From text embeddings
- **Formula:** $\text{Attention}(Q,K,V) = \text{softmax}(QK^\top/\sqrt{d})V$
- Allows the model to selectively focus on relevant text features for each spatial location

Classifier-Free Guidance

To strengthen alignment with text:

- Generate both conditional and unconditional latent representations
- Combine them using guidance scale:

$$z_{\text{guided}} = z_{\text{unconditional}} + \text{guidance\_scale} \cdot (z_{\text{conditional}} - z_{\text{unconditional}})$$

- Higher guidance scale leads to stronger adherence to text but potentially less diversity

7.5 Key Visualizations and Examples

Perceptual vs. Semantic Compression

The slides show a rate-distortion curve (page 14) illustrating:

- **Left side:** High distortion region (semantic compression) where LDMs operate
- **Right side:** Low distortion region (perceptual compression) where traditional autoencoders operate
- **Key Insight:** LDMs work because they focus on preserving semantic content rather than exact pixel values

Diffusion Process Visualization

Pages 15–16 illustrate the diffusion process:

- **Forward process:** Original dog image gradually becomes noise
- **Reverse process:** Denoising U-Net iteratively reconstructs image from noise

Text-to-Image Examples

Pages 23–26 show examples of text prompts creating different images:

- "Cat in Nyhavn" with sunset conditions
- "Dog in Nørrebro streets" with different lighting
- "Winter! Let it snow"
- "A dog on Mars" showing the creative capabilities of the model

7.6 Exam-Style Questions with Answers

Question 1: Why is operating in latent space more efficient than pixel space for diffusion models?

Answer: Diffusion in pixel space requires operating on all pixels (e.g., a 512×512 image has $\sim$786K dimensions), making it computationally expensive. Latent diffusion models first compress images to a much smaller latent representation (e.g., $64 \times 64 \times 4 \approx 16$K dimensions), reducing computational complexity by orders of magnitude. Additionally, the latent space primarily captures semantically meaningful information while discarding imperceptible details, making the diffusion process more efficient without sacrificing quality.

Question 2: Explain how CLIP enables text-to-image generation models.

Answer: CLIP enables text-to-image generation through several mechanisms:

1. It provides a text encoder that converts prompts into meaningful embeddings representing visual concepts
2. These embeddings are injected into the diffusion model's denoising process via cross-attention mechanisms
3. CLIP creates a shared embedding space between text and images, allowing the model to "understand" how text descriptions correspond to visual elements
4. CLIP's contrastive training on millions of image-text pairs provides rich knowledge of visual concepts described in natural language

Question 3: What is cross-attention and why is it important in Latent Diffusion Models?

Answer: Cross-attention is a mechanism where the query vectors come from one domain (spatial features in the diffusion U-Net) and the key-value pairs come from another domain (text embeddings). In LDMs, cross-attention allows text information to influence the spatial features during the denoising process. The attention operation is computed as:

$$\text{Attention}(Q,K,V) = \text{softmax}(QK^\top/\sqrt{d})V$$

where Q represents spatial queries, and K,V come from text embeddings. This is crucial because it enables the precise spatially-localized influence of text concepts on the generated image, effectively translating text descriptions into visual elements at appropriate locations.

Question 4: What is classifier-free guidance and what trade-offs does increasing the guidance strength present?

Answer: Classifier-free guidance is a technique that enhances text-image alignment by combining conditional and unconditional generation. The model produces both a text-conditioned latent representation and an unconditional one, then combines them using:

$$z_{\text{guided}} = z_{\text{unconditional}} + \text{guidance\_scale} \cdot (z_{\text{conditional}} - z_{\text{unconditional}})$$

Trade-offs when increasing guidance strength:

- **Higher guidance:** Stronger adherence to text prompts, more accurate concept depiction
- **Higher guidance:** Less diversity, more stereotypical representations
- **Higher guidance:** Potentially less realistic images with exaggerated features
- **Lower guidance:** More creative and diverse outputs but possibly less faithful to the text prompt

7.7 Advanced Topics Often Overlooked

Memory and Efficiency Advantages

The slides highlight the fundamental efficiency advantage of LDMs:

- Reduction of dimensions from $\sim$786K (pixel space) to $\sim$16K (latent space)
- This allows processing of higher resolution images
- Enables faster training and inference
- Requires less memory during processing

Text-Image Alignment Challenges

Several challenges arise in aligning text and images:

- Ambiguity in language (e.g., "a cat on a mat" has many valid visual interpretations)
- Abstract concepts are difficult to visualize consistently
- Bias in training data affects how concepts are represented
- Compositional challenges (correctly arranging multiple objects mentioned in text)

Advanced Conditioning Techniques

Beyond basic text conditioning:

1. **Textual Inversion** (slides 51–56): Learning custom token embeddings to represent specific concepts from just a few images
2. **DreamBooth** (slide 47): Fine-tuning the entire model to learn a specific subject
3. **ControlNet** (slide 48): Adding conditioning through pose, edges, or other structural information
4. **Prompt-to-Prompt** (slide 49): Precisely editing specific parts of generated images by manipulating cross-attention maps
5. **Instance Diffusion** (slide 50): Instance-level control for multiple objects in a scene

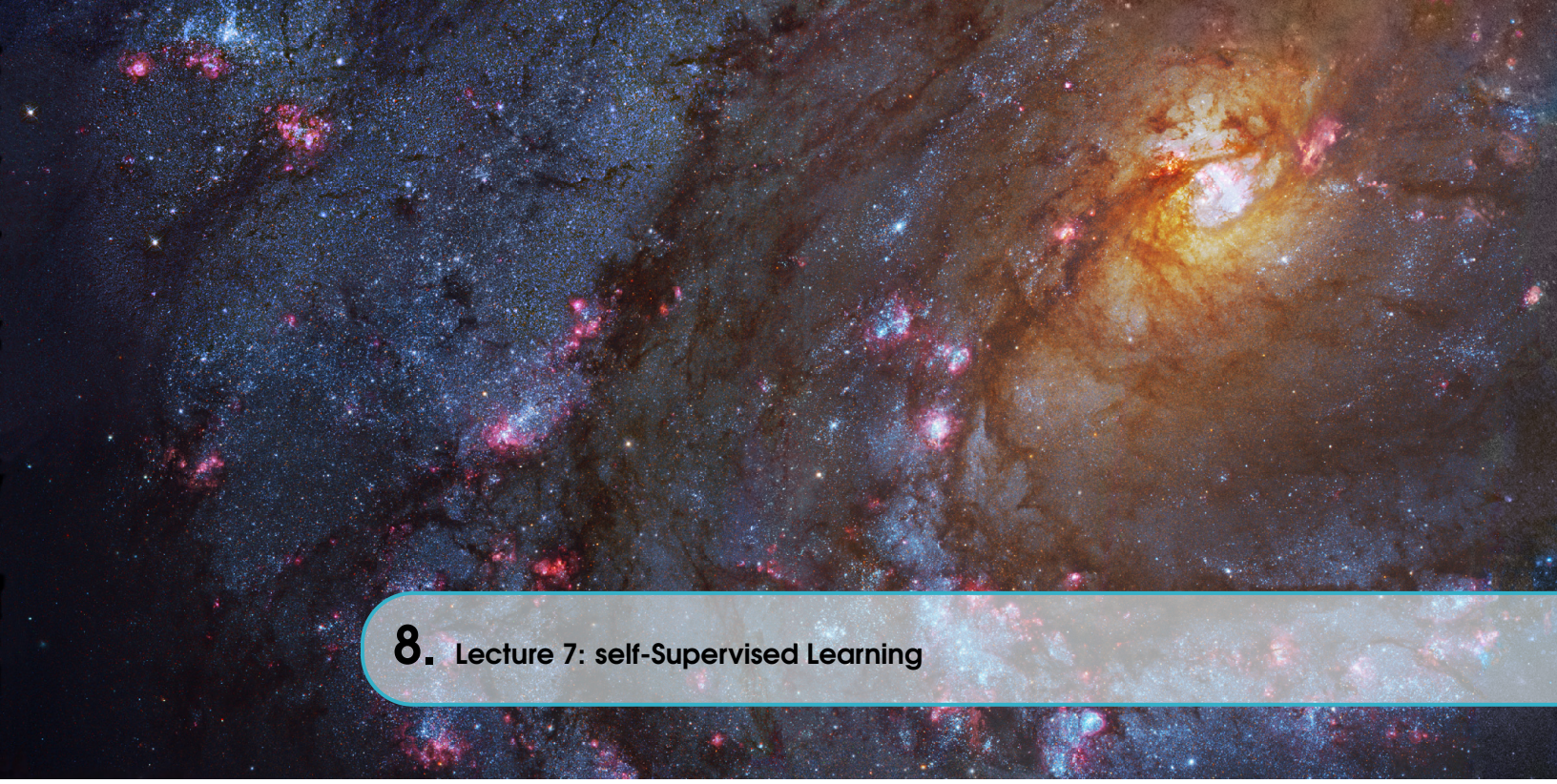
Balancing Generation Control and Freedom

The slides mention "CG/CFG is cool, but I want freedom" (slides 21–26):

- Highlights tension between control and creative freedom

- More guidance gives better text alignment but less diverse results
- Finding the right balance depends on the specific application
- Different control mechanisms provide different types of control (global vs. local)

This comprehensive overview covers the essential concepts, mechanisms, and nuances that would be expected in an oral exam on text-to-image generative AI models, with special focus on Latent Diffusion Models and CLIP.



8. Lecture 7: self-Supervised Learning

8.1 Why Self-Supervised Learning?

Self-supervised learning (SSL) emerged as a solution to two fundamental challenges in deep learning:

1. **Data hunger**: Deep learning models require massive amounts of data to perform well. The slides show numerous large datasets (ImageNet, COCO, Open Images, CityScapes, ADE20K, etc.) that power modern vision systems.
2. **Annotation bottleneck**: Human annotation is:
    - Expensive (slides show "Money??" with stacks of cash)
    - Time-consuming (20–40s per bounding box)
    - Not scalable for complex tasks (1.5–2 hours per image for panoptic segmentation)

The Key Insight

Self-supervised learning leverages the intrinsic structure within unlabeled data to create supervisory signals, enabling models to learn useful representations without human annotations.

Learning Paradigms Comparison

From the slides, we can clearly distinguish three approaches:

Supervised Learning	Unsupervised Learning	Self-Supervised Learning
Data: (x, y) pairs where x is data, y is label	**Data:** x (just data, no labels)	**Data:** Uses raw data but creates its own supervision
Goal: Learn a function to map $x \to y$	**Goal:** Learn underlying hidden structure	**Goal:** Learn representations that are useful for downstream tasks
Examples: Classification, object detection, segmentation	**Examples:** Clustering, dimensionality reduction	**Examples:** Pretext tasks like predicting image rotations
Limitation: Requires expensive annotations	**Limitation:** Representations might not be useful for specific tasks	**Advantage:** Combines best of both worlds

Table 8.1: Comparison of Learning Paradigms

The Self-Supervised Learning Process

Self-supervised learning (SSL) follows a two-step process:

1. Pretraining Stage

- Train a network on a "pretext task" that doesn't require human annotations
- The pretext task forces the model to learn useful visual representations

2. Downstream Application

- Transfer the pretrained encoder to target tasks
- Use methods like linear classifiers, KNN, or finetuning

Major Pretext Tasks

The slides categorize pretext tasks into three types:

1. Generative Tasks

Tasks that predict part of the input signal:

- **Autoencoders**: Reconstruct inputs, with variants including:
  - Sparse autoencoders
  - Denoising autoencoders
  - **Masked Autoencoders (MAE)**: Divide image into patches, mask most of them, and predict the masked patches
- **Colorization**: Convert grayscale images to color
- **Inpainting**: Remove parts of an image and train the model to reconstruct them

2. Discriminative Tasks

Tasks that predict properties about the input:

- **Context Prediction**: Predict relative location of patches from the same image (8 positions)
- **Rotation Prediction**: Predict the rotation applied to an image
- **Deep Clustering**:
  1. Randomly initialize CNN
  2. Extract features from many images
  3. Cluster features with K-means
  4. Use cluster assignments as pseudo-labels
  5. Repeat
- **Contrastive Learning**:
  - Identify which pairs of augmented images came from the same original image
  - Creates a similarity matrix between samples
  - Positive pairs (same image, different augmentations) $\rightarrow$ high similarity
  - Negative pairs (different images) $\rightarrow$ low similarity

3. Multimodal Tasks

Use additional signals besides RGB images:

- **Video**: Use temporal consistency between frames
- **Sound**: Match images with audio
- **3D**: Use depth information
- **Language**: Match images with text descriptions (like CLIP)

Influential Methods

DINO (Self-supervised Vision Transformers)

- Uses a student-teacher architecture
- Teacher parameters updated with EMA of student parameters
- Shows impressive results in semantic segmentation, depth estimation, and instance retrieval
- Produced by Facebook AI Research

CLIP (Contrastive Language-Image Pretraining)

- Uses natural language supervision
- Image and text encoders trained to predict which caption matches which image
- Enables zero-shot classification by comparing image embeddings with text embeddings of class names

Architecture Components

Encoders

- Networks that extract features from input data
- Can be CNNs or Vision Transformers
- The valuable part that gets transferred to downstream tasks

Data Augmentations

- Critical for many SSL methods, especially contrastive learning
- Create different views of the same image: crops, color jitter, rotations

- Help models learn invariances to these transformations

Projection Heads

- Additional layers after the encoder
- Used to map features to a space where contrastive loss is applied
- Often discarded when transferring to downstream tasks

Performance Insights

The slides show that SSL approaches have made significant progress:

- Contrastive approaches like MoCo, SimCLR show major improvements over earlier methods
- SSL methods reach 60–70% of fully supervised performance
- SSL features can be competitive with ImageNet supervision for downstream tasks
- DINOv2 achieves state-of-the-art results on multiple benchmarks without any supervision

Alternative Approaches to Reduce Annotation Costs

- **Active Learning**: Intelligently select which samples to annotate.
- **Weakly Supervised Learning**: Use image-level labels instead of bounding boxes.
- **Human-Machine Collaboration**: Combine human expertise with machine efficiency.
  - Examples: Verification, center-clicking, extreme clicking
  - Google reported using extreme clicking for 300M bounding boxes internally

Oral Exam Practice Questions

Fundamental Concepts

Q1: What is self-supervised learning and how does it differ from supervised and unsupervised learning?

A: Self-supervised learning creates supervisory signals from unlabeled data itself. Unlike supervised learning, it doesn't require human annotations, and unlike unsupervised learning, it explicitly defines a pretext task with clear learning objectives rather than just discovering data structure.

Q2: Explain the two-step process in self-supervised learning.

A: First, a model is pretrained on a pretext task that doesn't require human annotations (like predicting image rotations or missing patches). Second, the pretrained encoder is transferred to downstream tasks by attaching task-specific heads and using methods like linear probing or fine-tuning.

Pretext Tasks

Q3: Name and explain three categories of pretext tasks in self-supervised learning.

A:

1. **Generative tasks:** Predict parts of the input (e.g., autoencoders, colorization)
2. **Discriminative tasks:** Predict properties about the input (e.g., rotation, context prediction, contrastive learning)
3. **Multimodal tasks:** Use additional signals beyond RGB images (e.g., video, sound, text, 3D)

Q4: How does contrastive learning work in self-supervised learning?

A: Contrastive learning creates different augmentations of the same images, then trains the model to pull representations of augmentations from the same image closer together (positive pairs) while pushing representations from different images apart (negative pairs). This typically uses a similarity-based loss function.

Technical Details

Q5: What is the role of data augmentation in self-supervised learning?

A: Data augmentation creates different views of the same instance, allowing the model to learn invariances to transformations that don't change semantic content. This helps build robust representations that focus on important image properties rather than superficial details.

Q6: Explain the concept of a "pretext task" and provide two examples.

A: A pretext task is an artificial task designed to force the model to learn useful representations without human annotations. Examples include predicting the relative position of image patches (context prediction), reconstructing masked image patches (MAE), or identifying which augmented images came from the same original image (contrastive learning).

Methods and Models

Q7: How does DINO work and what makes it effective?

A: DINO uses a student-teacher architecture where the student processes two different augmentations of an image, and the teacher processes another. The student is trained to match the teacher's output distributions, while the teacher is updated using an exponential moving average (EMA) of the student's parameters. This self-distillation process produces features with emergent properties like semantic segmentation.

Q8: What is CLIP and how does it enable zero-shot classification?

A: CLIP (Contrastive Language-Image Pretraining) trains image and text encoders to predict which captions match which images in a dataset. For zero-shot classification, it encodes class names as text (e.g., "a photo of a dog") and compares image embeddings with these text embeddings to predict the class.

Performance and Applications

Q9: How do self-supervised methods compare to supervised methods in terms of performance?

A: Self-supervised methods have significantly improved, with current methods reaching 60–70% of fully supervised performance. For some specific tasks like semantic segmentation, methods like DINO can achieve competitive results without any supervision. The gap continues to narrow with newer methods.

Q10: Why is self-supervised learning particularly important for computer vision?

A: Computer vision tasks typically require large amounts of labeled data that is expensive and time-consuming to create (e.g., 1.5–2 hours per image for panoptic segmentation). Self-supervised learning allows models to learn from vast amounts of unlabeled images, reducing dependence on annotations and enabling more generalizable representations.

Commonly Overlooked Details

- **Feature collapse problem:** Without proper constraints, self-supervised models might learn to produce constant or trivial features. Methods use various techniques to prevent this, like

contrastive loss, stop-gradient operations, or batch normalization.

- **Projection heads:** Many methods use a projection head on top of the encoder during pretraining but discard it for downstream tasks. This separation between pretraining and representation spaces is important for performance.
- **Teacher-student asymmetry:** In methods like DINO, creating asymmetry between teacher and student networks (through EMA updates, centering, or temperature differences) is crucial for preventing representation collapse.
- **Transfer gap:** The performance on pretext tasks doesn't always correlate perfectly with downstream task performance. Good pretext tasks should align with properties needed for downstream applications.

9. Lecture 8: Multi-Modal Learning

Core Concepts and Motivations

Multimodal learning is a fundamental approach in AI that integrates information from multiple modalities (like vision, text, audio) to build systems that understand and process information more comprehensively, similar to how humans perceive the world through diverse senses.

Key Motivations

- Humans naturally perceive the world through multiple senses simultaneously (vision, hearing, touch, smell, taste, balance)
- Single modality systems miss contextual information that exists across modalities
- Different modalities provide complementary information that leads to richer understanding
- Real-world applications often require processing multiple input types concurrently

Main Modalities Covered

- **Visual:** images, videos
- **Textual:** natural language
- **Audio:** speech, sounds
- **Kinesthetic:** motion, position

Multimodal Tasks and Applications

Vision-Language Tasks

1. **Visual Question Answering (VQA)**
    - **Input:** Image + text question
    - **Output:** Text answer about visual content
    - **Example:** "What is the dog holding with its paws?" $\rightarrow$ "Frisbee"
    - Can be multiple-choice or open-ended answers
2. **Image Captioning**

- **Input:** Image
- **Output:** Generated text description (sentence or paragraph)
- **Process:** (1) Understanding visual content, (2) Encoding into representation, (3) Generating coherent description

3. **Image-Text Retrieval**
   - **Tasks:** Text-to-image retrieval and image-to-text retrieval
   - **Approach:** Embedding images and text in shared semantic space
   - **Applications:** Finding images that match text queries or captions for images
4. **Visual Grounding**
   - **Input:** Image + text phrase/query
   - **Output:** Bounding box/region in the image
   - **Types:** Phrase grounding and referring expression comprehension
   - **Example:** "The red frisbee next to the dog" $\rightarrow$ highlights object in image
5. **Text-to-Image Generation**
   - Uses text prompts to condition generative models
   - Often employs diffusion models in latent space
   - Requires strong semantic alignment between modalities

Video Understanding

- Combines spatial information (what's in a frame) with temporal information (how things change over time)
- Two-stream convolutional networks process:
  - **Spatial stream:** Recognizes objects and scenes in individual frames
  - **Temporal stream:** Captures motion via optical flow between frames

Key Architectures and Models

Evolution of Architectures

Early Approaches:

- CNN+LSTM architectures (CNN extracts image features, LSTM processes text)
- Separate encoders with simple fusion mechanisms

Modern Approaches:

- Transformer-based architectures with sophisticated fusion strategies
- Joint embedding spaces with multimodal attention mechanisms
- Large language models enhanced with visual processing capabilities

Important Models and Their Innovations

1. CLIP (Contrastive Language-Image Pre-training)

- Trains on image-text pairs using contrastive loss
- Each image predicts which caption matches from a batch
- Creates aligned visual and textual embeddings in shared space
- Enables zero-shot transfer to many downstream tasks

2. ViLT (Vision-and-Language Transformer)

- Eliminates need for separate object detection or region features
- Directly processes image patches and text tokens

- Uses transformer encoder with multimodal attention
- Trained with image-text matching, masked language modeling, and word-patch alignment

3. BLIP (Bootstrapping Language-Image Pre-training)

- Combines three components:
  - Unimodal encoder (image-text contrastive loss)
  - Image-grounded text encoder (image-text matching)
  - Image-grounded text decoder (language modeling for caption generation)
- Synthesizes and filters noisy image-text pairs during training

4. Grounding DINO

- Extends closed-set object detectors to open-vocabulary scenarios
- Combines vision transformer with text encoder
- Feature fusion between text and image at multiple levels
- Can process both predefined categories and natural language queries

5. LLaVa and Multimodal LLMs

- Vision encoder + projection layer + language model
- Projects visual features into language model's embedding space
- Preserves LLM's generative capabilities while adding visual understanding
- Represents the current trend of extending LLMs to multimodal contexts

Fusion Strategies

The slides demonstrate several approaches to combining information across modalities:

1. **Early Fusion**
   - Combines raw or lightly processed inputs before main processing
   - Allows deep interaction between modalities throughout the network
   - *Challenge*: Modalities may have very different statistical properties
2. **Late Fusion**
   - Processes each modality separately and combines high-level features/predictions
   - *Example*: Two-stream networks for video that fuse spatial and temporal outputs
   - Simpler but may miss cross-modal interactions
3. **Multi-level Fusion**
   - Combines information at multiple stages in the processing pipeline
   - *Example*: Grounding DINO with feature fusion at backbone, neck, and head levels
4. **Cross-Attention Mechanisms**
   - Allows each modality to attend to relevant parts of the other
   - Common in transformer-based architectures like BLIP and VILT
   - Enables fine-grained alignment between modalities
5. **Joint Embedding Space**
   - Maps different modalities to a common representational space
   - Enables direct comparison and retrieval across modalities
   - Trained with contrastive or alignment objectives (e.g., CLIP, BLIP)

Pre-training Objectives and Downstream Tasks

Common Pre-training Objectives:

- Masked Language Modeling

- Masked Image Modeling
- Image-Text Matching (is this text related to this image?)
- Image-Text Contrastive Learning (align similar pairs in embedding space)

Connection to Downstream Tasks:

- These pre-training objectives prepare models for specific multimodal applications
- Different objectives benefit different downstream tasks
- Most modern approaches use multiple objectives simultaneously

Oral Exam Quiz Questions

Basic Understanding

1. **Q: What is multimodal learning and why is it important?**
   A: Multimodal learning involves AI systems that can process and integrate information from multiple types of inputs (modalities) such as vision, text, and audio. It's important because real-world information comes in multiple forms, humans perceive the world through multiple senses, and integrating these modalities leads to richer understanding and more capable AI systems.
2. **Q: Explain the difference between early fusion and late fusion in multimodal models.**
   A: Early fusion combines raw inputs or low-level features from different modalities before main processing, allowing deep interaction but potentially struggling with different statistical properties. Late fusion processes each modality separately with specialized networks and only combines high-level features or predictions, which is simpler but might miss important cross-modal interactions.

Architecture and Models

3. **Q: How does CLIP train image and text embeddings to align in a shared space?**
   A: CLIP uses contrastive learning where each image in a batch is paired with its corresponding text description. The model is trained to maximize similarity between matching image-text pairs while minimizing similarity with non-matching pairs. This creates a joint embedding space where semantically similar images and texts are close to each other, enabling cross-modal retrieval and zero-shot classification.
4. **Q: What innovative approach does VILT take compared to earlier vision-language models?**
   A: VILT (Vision-and-Language Transformer) eliminates the need for region-based features or separate object detection by directly processing image patches alongside text tokens in a unified transformer architecture. It uses simple linear projection of flattened patches rather than convolutional processing, making it more efficient while maintaining strong performance on vision-language tasks.

Multimodal Applications

5. **Q: Compare and contrast Visual Question Answering (VQA) and Visual Grounding.**
   A: Visual Question Answering takes an image and a question as input and produces a textual answer based on understanding the visual content. Visual Grounding takes an image and a textual description and produces spatial coordinates (like a bounding box) locating the

described object/region in the image. VQA requires image understanding to generate language, while grounding requires language understanding to locate visual elements.

6. **Q: How do two-stream networks process videos, and why is this approach effective?**
   A: Two-stream networks use separate pathways: a spatial stream processing individual frames to understand objects/scenes, and a temporal stream processing optical flow to capture motion. This is effective because it separates the "what" (object recognition) from the "how" (motion analysis), mirroring human visual processing and allowing specialized architectures for each aspect before combining their insights.

Advanced Concepts

7. **Q: How do Multimodal Large Language Models (MLLMs) like LLaVa integrate visual information with language models?**
   A: MLLMs like LLaVa use a vision encoder (like a pre-trained ViT) to extract visual features, which are then transformed by a projection layer to map them into the embedding space of the language model. This allows the language model to process visual tokens alongside text tokens, enabling it to reason about and generate text based on both visual and textual inputs without significant architectural changes to the core LLM.
8. **Q: What challenges arise in training multimodal systems, and how are they typically addressed?**
   A: Challenges include:
   - (1) Different learning dynamics across modalities, addressed through careful initialization or freezing pre-trained components
   - (2) Modality imbalance, addressed through balancing loss terms or gradient manipulation
   - (3) Alignment between different semantic spaces, addressed through contrastive learning or joint embedding objectives
   - (4) Data requirements, addressed through large-scale pre-training and data augmentation
   - (5) Computational complexity, addressed through efficient architectures and training strategies

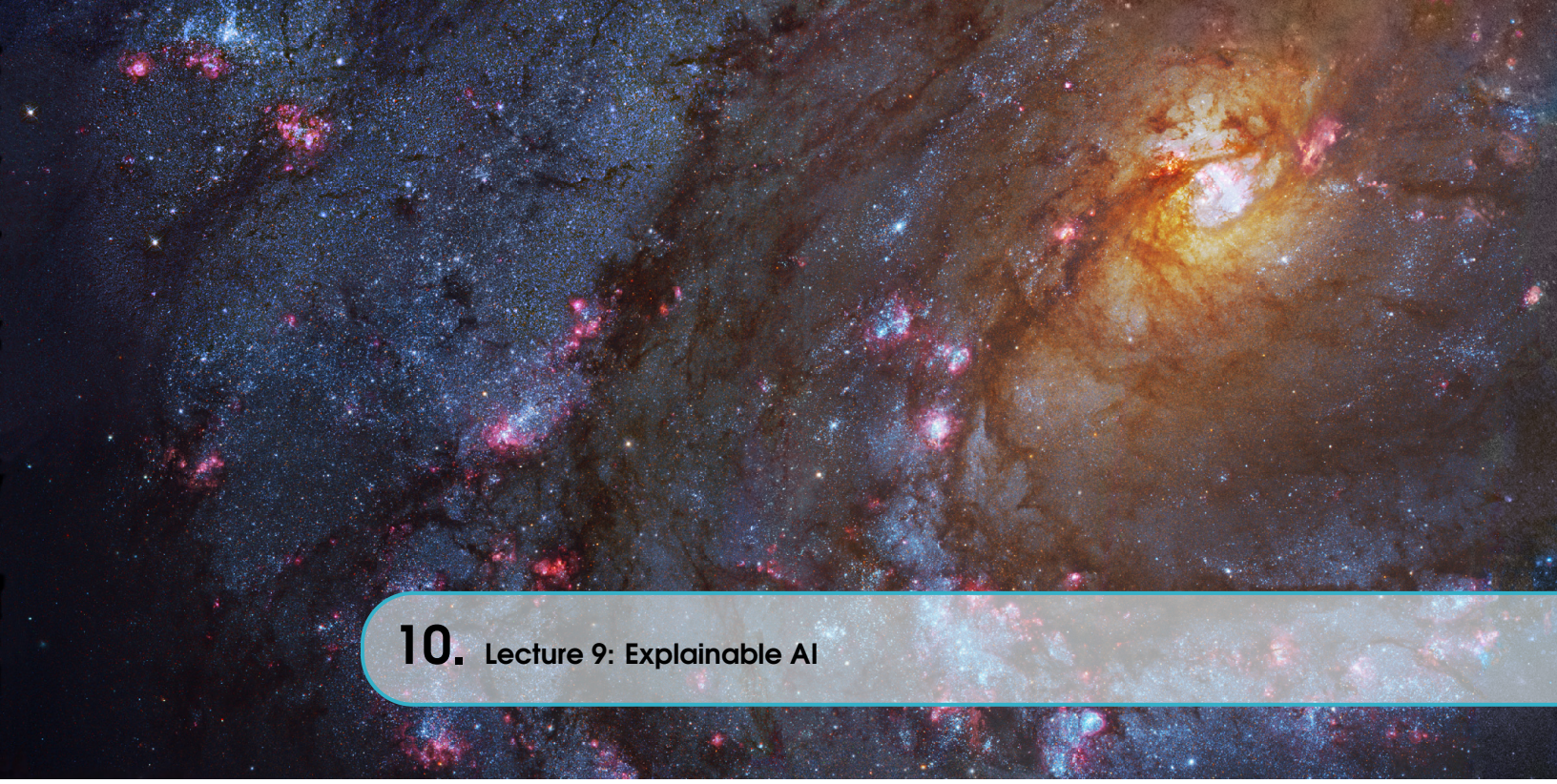
Expert-Level Insights and Subtleties

1. **Modality Gaps and Alignment**
   - Different modalities have fundamentally different statistical properties and semantics
   - Creating truly aligned representations remains challenging
   - Contrastive learning has emerged as a powerful approach but doesn't guarantee perfect alignment
2. **Transfer and Generalization**
   - Models like CLIP show remarkable zero-shot transfer capabilities
   - Pre-training on diverse, large-scale multimodal data enables generalization
   - The quality and diversity of training data often matters more than architectural choices
3. **Evolution Toward Multimodal Foundation Models**
   - Trend from task-specific architectures to general-purpose multimodal foundations
   - Recent shift toward extending LLMs with multimodal capabilities rather than building custom architectures
   - Emergence of multimodal transformers that handle multiple modalities natively
4. **Open vs. Closed Vocabulary Understanding**

- Earlier systems were limited to predefined categories (closed-set)
- Modern approaches enable open-vocabulary understanding through language guidance
- Models like Grounding DINO show how language can extend visual systems to novel concepts

5. **Cross-Modal Generation**
   - Beyond understanding, models can now generate one modality from another
   - Text-to-image models like those using diffusion processes show remarkable generative capabilities
   - These rely on the alignment between modalities established during pre-training

Being able to discuss these more nuanced aspects shows deeper understanding of multimodal learning that would impress in an oral exam setting.



10. Lecture 9: Explainable AI

Explainable AI (XAI) in Computer Vision: Exam Preparation Guide

Core Concepts and Motivations

What is XAI?

XAI focuses on answering *"What caused my prediction?"* — making AI decision-making processes transparent and understandable to humans. The lecture by Aasa Feragen outlines several approaches to this challenge, specifically in computer vision.

Key motivations for XAI:

- **Utility and user trust** – predictions are more useful when users understand them
- **Accountability** – decisions that cannot be explained may not be suitable for critical applications

The counterfactual perspective: A significant portion of the lecture focuses on counterfactual explanations, which answer the question: *"How should I change the input to obtain a different classification?"* This represents a key paradigm in XAI for computer vision.

XAI Methods and Techniques

1. Saliency-Based Methods

Saliency maps:

- Create heatmaps highlighting regions of the image most important for predictions
- **Example:** Medical image classification (slide 7) showing pneumonia detection

Vanilla gradient saliency:

- Computes the gradient of output with respect to input pixels
- **Limitation:** Tends to be "noisy" due to classifiers not being smooth over images

SmoothGrad:

- Improves saliency maps by adding Gaussian noise to the image
- Ironically "removes noise by adding noise"
- Creates more coherent, interpretable heatmaps

2. Feature Attribution Methods

Class Activation Mapping (CAM):

- Post-hoc method that identifies important features via activation analysis
- Maps activations back to input space to highlight relevant regions

2. Feature Attribution Methods (continued)

GradCAM:

- Activation mapping weights computed via gradients
- Uses ReLU to turn off negative contributions for more intuitive output
- **Example:** The cat/dog image (slide 14) showing how different features are highlighted

Attention-based explanations:

- The lecture notes that previous methods were post-hoc (added after model training)
- Attention mechanisms provide intrinsic explanations (built into the model)

3. Prototype and Concept-Based Methods

Retrieval-based explanation:

- Explains through examples of similar data with similar predictions
- Uses nearest neighbors in representation space

Concept bottleneck models:

- Uses predefined explanatory concepts that users can interact with
- Trains on input x, concept vectors c, and outputs y
- **Example:** Medical image classifier that explicitly detects concepts like "bone spurs" or "sclerosis"

Prototype-based explanation:

- Performs classification using similarity to learned prototypes
- **Example:** Bird classification by identifying similar prototype parts across images

4. Counterfactual Explanations

Diffusion Models for Counterfactuals:

- Diffusion model ensures counterfactuals stay on the data manifold
- Classification loss drives the counterfactual toward target class
- Perceptual loss keeps counterfactual visually similar to original image

Adversarial Counterfactual Visual Explanations:

- Uses adversarial attacks to create counterfactuals
- Applies diffusion model to project attacks back onto data manifold
- Post-processing with inpainting to maintain similarity to original

Applications of XAI

Detecting Shortcut Learning

- Using counterfactuals to identify when models learn spurious correlations
- **Example:** Medical imaging diagnosis based on artifacts rather than pathology
- Testing how changing shortcut features affects model predictions

Improving Image Quality

- Using diffusion models to enhance image quality
- **Example:** Diff-ICE method to improve ultrasound imaging

Attribution for Generative AI

- Assigning credit for creative works generated by AI
- **Example:** EKILA system that identifies training images that contributed to generated art

Interpreting Text-to-Image Models

- What the DAAM: Analyzing which parts of text prompts influence image regions
- Interpreting CLIP's image representation via text-based decomposition
- Hidden language of diffusion models: Conceptor approach

Quiz Questions and Answers

1. **What are the two main motivations for XAI discussed in the lecture?**
   A1: Utility/user trust (predictions are more useful when understood) and accountability (decisions that cannot be explained may be unsuitable for critical applications).
2. **How does SmoothGrad improve upon vanilla saliency maps?**
   A2: SmoothGrad adds Gaussian noise to the input image multiple times, computes gradients for each noisy sample, and averages them. This produces smoother, more interpretable heatmaps by "removing noise by adding noise."
3. **What is the difference between post-hoc and intrinsic explanation methods?**
   A3: Post-hoc methods (like saliency maps and GradCAM) add explanations after the model is trained, while intrinsic methods (like attention mechanisms) incorporate explainability directly into the model architecture.
4. **How do diffusion models contribute to counterfactual explanations?**
   A4: Diffusion models serve as the data compatibility term, ensuring counterfactuals remain on the data manifold (look realistic). They help generate plausible modifications that change the prediction while maintaining image coherence.
5. **What two main terms are typically included in counterfactual explanation methods?**
   A5: A data compatibility term (keeping the counterfactual close to realistic data) and a term that drives the counterfactual toward the desired prediction or class.
6. **How can prototype-based explanations help interpret image classifications?**
   A6: They identify which parts of the input image look similar to learned prototypes, providing an intuitive way to understand why an image belongs to a specific class (e.g., "this bird was classified as a sparrow because its beak looks like this prototype").
7. **What practical applications of XAI beyond model understanding were discussed?**
   A7: Detecting shortcut learning (when models learn spurious correlations), improving image quality, attribution for generative AI, and explaining text-to-image generation models.

Key Insights for the Oral Exam

Conceptual understanding is crucial:

- Be prepared to explain not just how XAI methods work technically, but why they're needed
- Understand the difference between explanation types (feature attribution vs. counterfactual)

Evolution of XAI methods:

- Early methods were post-hoc (saliency maps, GradCAM)

- More recent methods are intrinsic (attention mechanisms) or generative (diffusion-based)

The role of diffusion models:

- Appear throughout the lecture both as subjects of explanation and tools for generating explanations
- Represent an important connection between generative AI and explainability

Generative AI explainability:

- This is an emerging area with "not that much out there" (slide 29)
- Focus on techniques like DAAM, Conceptor, and prototype extraction

Multiple applications beyond understanding:

- XAI isn't just for transparency but can improve models and enable new capabilities
- Applications include detecting model biases, improving image quality, and fair attribution

The tension between performance and explainability:

- Complex models may perform better but be harder to explain
- Concept bottleneck models and prototype-based methods attempt to bridge this gap

Remember that the lecture emphasizes both traditional predictive model explanation and newer generative AI explanation. For the oral exam, be prepared to discuss how XAI techniques are evolving to address the growing complexity of AI systems, particularly in the domain of computer vision.

11. Lecture 10: Fairness

Algorithmic Fairness: Key Concepts for Your Oral Exam

Based on Professor Aasa Feragen's slides on Algorithmic Fairness, this guide summarizes key definitions, distinctions, and sources of bias.

1. Key Definitions and Distinctions

Core Fairness Concepts

- **Algorithmic fairness**: Aims to reduce unwanted bias by enforcing different notions of fairness
- **Three main fairness criteria:**
  - **Independence**: Equal acceptance rates across groups
  - **Separation**: Equal TPR, FPR, TNR, FNR across groups
  - **Sufficiency**: Equal positive/negative predictive value (i.e., given a prediction, equal probability it's true)

Understanding Bias

- **What bias is NOT**: Over/under-representation isn't inherently bias (e.g., breast cancer being more prevalent in women)
- **What bias IS**: "Data and algorithmic bias refers to systematic errors that differ between groups"
- **Types of bias:**
  - Outcome label bias (incorrect ground truth)
  - Representational bias (imbalanced training data)
  - Measurement bias (using inappropriate proxies)
  - Algorithmic amplification (models exaggerating existing patterns)

2. Case Studies of Algorithmic Bias

COMPAS Criminal Risk Assessment

- Racial bias in predicting risk of re-offense among US criminals

- ProPublica investigation showed disparate impact for Black defendants
- Proxy variable for criminality (previous verdicts) was itself biased

Google Photos Misclassification

- Algorithm incorrectly labeled Black people as gorillas
- Root cause: poor model robustness and uncertainty handling
- Demonstrates risks of using algorithms outside training domain

Healthcare Algorithm Bias

- Algorithm falsely concluded Black patients were healthier than equally sick White patients
- Problem: algorithm used healthcare costs as a proxy for health needs
- Less money spent on Black patients with same level of need created biased labels

Gender Bias in Medical Imaging

- Diagnostic accuracy for conditions like pneumothorax varied by gender
- Accuracy depended on gender representation in training data
- Sometimes complex: "I thought I knew the cause of bias – but it wasn't that simple!"
- Larrazabal et al. (PNAS 2020) showed the best model for women still performed better for men

Biases in Text-to-Image Generation Models

- Generated images reflect and sometimes amplify social stereotypes
- Occupational stereotypes: software developers shown as men, flight attendants as women
- Racial biases in depictions of "terrorists," "thugs," etc.
- Personality trait associations with gender (study by Girrbach et al., 2025)

3. Technical Approaches to Measure and Mitigate Bias

Bias Detection

- Visualizing disparities in error rates across groups
- Measuring differences in outcomes across protected attributes
- Using XAI to help identify sources of bias (DeGrave et al., 2021)

Debiasing Text-to-Image Models

Debiasing text embeddings:

- Orthogonal projection to remove biased directions
- Calibration with positive pairs (e.g., "a photo of a male doctor" $\approx$ "a photo of a doctor")

Soft prompt optimization:

- Finetuning just a few tokens (5) can significantly reduce gender bias

Model finetuning:

- Distributional alignment loss to steer outputs toward target distributions

Trade-offs in Debiasing

- Debiasing can sometimes backfire (Google Gemini example)

- Risk of overcorrection leading to historically inaccurate outputs
- Balance between fixing bias and preserving accuracy

4. Critical Insights and Reflections

Algorithms vs. Human Bias

- "Algorithms are new, bias is not"
- Algorithmic bias easier to detect than human bias
- AI/ML can potentially help identify existing biases in systems
- "Algorithms come with potential for early discovery of bias"

Context-Dependent Fairness

- No universal definition of fairness
- Appropriateness of fairness criteria depends on context:
  - CV screening with photos
  - Face recognition
  - Diagnosing cancer

Complex Sources of Bias

- Multiple potential sources: data imbalance, proxy variables, label errors
- Sometimes the causes aren't obvious
- Need for comprehensive approach to bias mitigation

Generative AI Challenges

- More complex fairness considerations than predictive models
- "So many more ways for bias to take effect – harder to measure"
- Open question of how to generalize fairness definitions to generative AI

5. Exam-Style Questions and Answers

Q1: What are the three main notions of algorithmic fairness, and how do they differ?

A1: The three main notions are:

- **Independence:** Equal acceptance rates across groups (regardless of true outcomes)
- **Separation:** Equal error rates (TPR, FPR, TNR, FNR) across groups
- **Sufficiency:** Equal predictive values (given a prediction, equal probability it's true)

They differ in what conditional probabilities they equalize and which aspects of fairness they prioritize.

Q2: How did bias manifest in the healthcare algorithm case study?

A2: The algorithm used healthcare costs as a proxy for health needs. Since historically less money was spent on Black patients with the same level of need, the algorithm falsely concluded Black patients were healthier than equally sick White patients, reducing the number of Black patients identified for extra care by more than half.

Q3: Why is representation in training data important for model fairness?

A3: Poor representation leads to poor performance for underrepresented groups. The Larrazabal study showed that diagnostic accuracy for pneumothorax in chest X-rays varied with gender representation in the training data. Models trained predominantly on men's X-rays performed worse on women, and vice versa.

Q4: What is the difference between bias in predictive AI and generative AI?

A4: While both have similar sources of bias (training data, proxies, etc.), generative AI presents more challenges because:

- There are many more ways bias can manifest
- It's harder to measure and quantify bias in generated content
- The output space is much larger and more subjective
- The relationship between bias and societal harm may be more complex

Q5: How can biased proxy variables lead to algorithmic discrimination?

A5: Biased proxies create biased labels, which produce biased models. In the healthcare case, using costs as a proxy for health needs incorporated historical spending disparities. In COMPAS, using previous verdicts as a proxy for criminality incorporated historical judicial biases. These proxies encode historical discrimination into the algorithm's predictions.

Q6: Explain why algorithmic bias can sometimes be easier to detect than human bias.

A6: Algorithms make systematic, consistent decisions that can be analyzed across many cases. We can directly measure performance differences across groups, visualize disparities, and identify patterns that might be missed in individual human decisions. Because algorithms are built on data, we can also analyze that data before deployment to detect potential issues.

6. Take-Away Messages

As summarized in the final slide:

1. Using AI outside the training domain is risky
2. Bias in data gives bias in models
3. Algorithmic bias is easy to detect (at least in predictive models)
4. You can get biased models even if you "do it all right"
5. Debiasing is nontrivial and not well defined
6. Across the board: Be cautious, react to odd AI behavior

Good luck with your exam! These concepts demonstrate how algorithmic fairness sits at the intersection of technical implementation, ethical considerations, and societal impact.

12. Exercise 1

ADLCV Exercise 1: Implementing a Transformer for Sentiment Classification

I'll help you understand the theory and implementation details of this transformer exercise for sentiment classification on IMDB reviews.

1. Exercise Overview

Based on the provided documents, the exercise focuses on building a transformer model for sentiment classification of IMDB movie reviews. The main tasks are:

1. Complete the attention module implementation
2. Complete the positional encoding implementation
3. Train and evaluate the model on IMDB reviews
4. Analyze how different model choices affect performance
5. Optional: Implement a CLS token pooling strategy
6. Optional: Implement learnable positional encoding
7. Optional: Fine-tune a pre-trained BERT model

2. Core Concepts for Understanding Transformers

Transformer Architecture

The transformer model in this exercise follows the original architecture from *"Attention Is All You Need"* with some simplifications for classification:

- **Encoder-only design**: Unlike the original encoder-decoder architecture for translation
- **Token embeddings**: Convert discrete tokens to continuous vector representations
- **Positional encoding**: Add position information (since transformers have no inherent sequence awareness)
- **Multi-head attention**: Parallel attention mechanisms for capturing different aspects of the input

- **Encoder blocks**: Combining attention, normalization, and feedforward layers with residual connections
- **Classification head**: Final layer to convert sequence representations to sentiment prediction

Self-Attention Mechanism

The attention mechanism is the core innovation of transformers:

- Computes interactions between all pairs of tokens in the sequence
- Each token can "attend" to all other tokens with different weights
- The mechanism uses queries (Q), keys (K), and values (V) projections
- Attention weights are computed as: $\text{softmax}(QK^\top/\sqrt{d_k})$
- Multi-head attention runs multiple attention mechanisms in parallel

Positional Encoding

Since transformers process all tokens simultaneously (not sequentially):

- Positional information must be explicitly added
- **Fixed sinusoidal encoding**: Uses sine/cosine functions of different frequencies
- Each position gets a unique encoding that follows mathematical patterns
- **Learnable encoding**: Position embeddings learned during training

Understanding Attention Mechanism

Q: Why is self-attention crucial for transformers?

A: Self-attention allows each token to directly attend to all other tokens in the sequence, capturing long-range dependencies that would be difficult with RNNs. It enables parallel processing (unlike sequential RNNs) and creates context-aware representations by weighting the importance of different tokens for each position.

Q: What's the purpose of multi-head attention?

A: Multi-head attention allows the model to:

- Attend to information from different representation subspaces
- Capture different types of relationships (syntactic, semantic, etc.)
- Increase model's representation power
- Learn patterns at different positions simultaneously

Positional Encoding

Q: Why do transformers need explicit positional encoding?

A: Unlike RNNs, transformers process all tokens in parallel with no inherent notion of sequence order. Positional encodings inject information about token positions, allowing the model to consider the order of words, which is crucial for language understanding.

Q: Explain the advantages of sinusoidal positional encoding.

A: Sinusoidal encoding:

- Provides a unique pattern for each position
- Allows the model to generalize to sequence lengths not seen during training

- Creates smooth transitions between positions
- Enables the model to easily compute relative positions through linear combinations

Model Architecture Choices

Q: How do different pooling strategies affect classification performance?

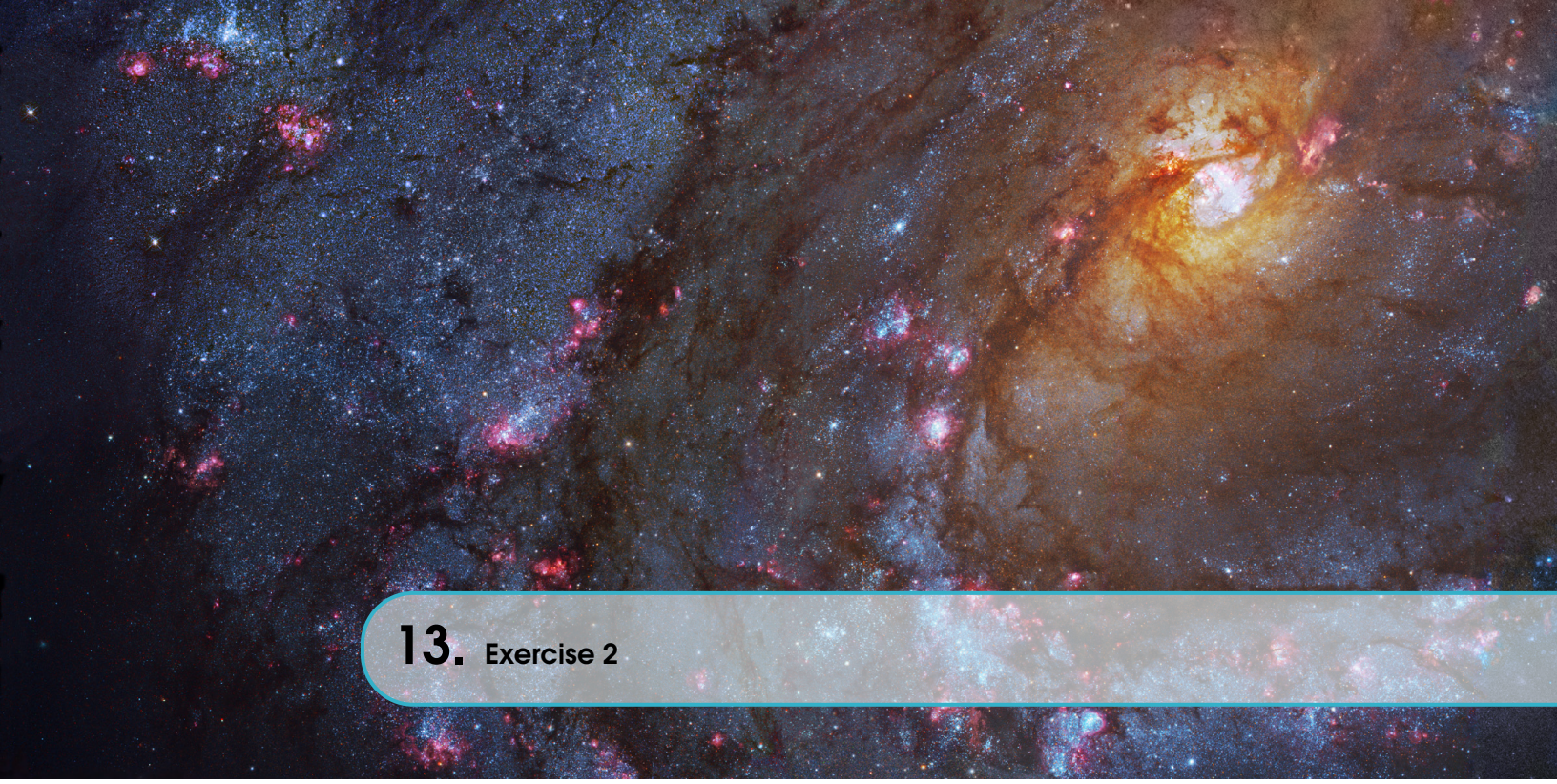
A:

- Mean pooling averages all token representations, treating all tokens equally
- Max pooling captures the most salient features across the sequence
- CLS token pooling learns a special token that aggregates sequence information
- The best choice depends on the task: CLS often works well for classification when properly trained

Q: How does the number of layers affect transformer performance?

A: More layers:

- Increase model capacity and ability to learn complex patterns
- Allow hierarchical feature learning
- May lead to overfitting on smaller datasets
- Require more computational resources
- Can face optimization challenges (vanishing/exploding gradients)



13. Exercise 2

Vision Transformers (ViTs) for Image Classification: Exercise 1.2 Guide

1. Conceptual Foundations of Vision Transformers

The Patching Mechanism

Intuitive Explanation: In CNNs, we process images using sliding windows (convolutions) that gradually build up features. ViTs take a completely different approach – they simply cut the image into a grid of non-overlapping patches, similar to dividing a photo into equal squares. It's like converting an image into a "sequence of image words" that can be processed by a transformer.

Technical Summary: The patching mechanism divides an input image of shape (H, W, C) into

$$N = \frac{H \times W}{P \times P}$$

non-overlapping patches, each of size (P, P, C), where P is the patch size. This eliminates the inductive bias of local spatial relationships inherent in CNNs, allowing the model to learn relationships between any patches regardless of spatial distance.

Patch Embeddings

Intuitive Explanation: After cutting an image into patches, we need to convert each patch into a format the transformer can understand – a vector of numbers. The patch embedding is like a translator that converts raw pixel values into meaningful feature representations.

Technical Summary: Patch embeddings are implemented as a linear projection that maps each flattened patch of dimension $P^2 \times C$ to an embedding dimension D. This is typically implemented as a single linear layer with weight matrix of shape $(P^2 \times C, D)$. This is analogous to token embeddings in NLP transformers and serves as the first stage of feature extraction.

Transformer Encoder Blocks

Intuitive Explanation: Each transformer block allows patches to "communicate" with each other (via self-attention) and then "think" about what they've learned (MLP). Layer normalization keeps

values in a reasonable range, while residual connections allow information to flow directly from earlier layers.

Technical Summary:

- **Self-attention:** Allows each patch to attend to all other patches using query-key-value projections.
- **MLP (Feed-forward Network):** Two-layer network (expansion factor typically $4\times$) with GELU activation.
- **Layer Normalization:** Stabilizes training by normalizing features.
- **Residual Connections:** Connect input and output of each sub-block to mitigate vanishing gradients.

Positional Encoding

Intuitive Explanation: Since transformer attention has no built-in sense of space or order, we need to explicitly tell it where each patch comes from in the original image. Positional encoding is like adding spatial coordinates to each patch.

Technical Summary:

- **Sinusoidal (Fixed) Encoding:** Uses sine and cosine functions of different frequencies
- **Learnable Encoding:** Initialized randomly and optimized during training
- Both are added directly to patch embeddings before being fed to transformer blocks

Attention Map Visualization

Intuitive Explanation: Attention maps show which parts of an image the model is focusing on when making decisions. They reveal whether the model is looking at relevant image regions or getting distracted.

Technical Summary: Attention maps represent the weight matrices from self-attention, showing how much each patch attends to every other patch. Different attention heads often specialize in different semantic aspects, which can be visualized as heatmaps overlaid on the original image.

ViT vs CNNs

Intuitive Explanation: CNNs are like experts focused on local details who gradually build up to seeing the bigger picture. ViTs are like generalists who immediately consider relationships between all parts of an image. This makes ViTs potentially more powerful but also more data-hungry.

Technical Summary:

- **Inductive Biases:** CNNs have strong biases (locality, translation equivariance). ViTs have minimal inductive biases.
- **Data Efficiency:** CNNs typically require less data due to their built-in assumptions about images.
- **Global Context:** ViTs capture global dependencies from the first layer through self-attention.
- **Computational Complexity:** Self-attention scales quadratically with sequence length, while convolutions scale linearly with image size.

ViT Architecture Structure

The Vision Transformer in `vit.py` follows this structure:

1. **Input Layer:** Images $\rightarrow$ patches $\rightarrow$ embedded tokens
2. **Class Token:** Optional prepended learnable token for classification
3. **Positional Embeddings:** Added to token embeddings

4. **Transformer Blocks:** Series of self-attention and MLP layers with residuals
5. **Pooling Layer:** Aggregates information (CLS token, mean, or max pooling)
6. **Classification Head:** Final linear projection to class logits

The encoder blocks follow the standard transformer architecture with self-attention, MLP, layer normalization, and residual connections.

Training and Evaluation Framework

The `imageclassification.py` implements:

1. **Data Preparation:** Loads CIFAR10, selects two classes (cat and horse by default)
2. **Model Initialization:** Configures a ViT with appropriate parameters
3. **Training Loop:** Uses AdamW optimizer with learning rate warmup
4. **Evaluation:** Computes validation accuracy and loss after each epoch

Effect of Key Hyperparameters

Parameter	Effect
Patch Size	Smaller (2x2): More detail but higher computation. Larger (8x8): More efficient but less detail.
Embedding Dim	Larger: More capacity but potential overfitting. Smaller: Fewer parameters but limited capacity.
Number of Heads	More heads: Capture diverse relationships. Fewer heads: Simpler with fewer parameters.
Number of Layers	Deeper models: More capacity but harder to train. Shallower models: Less overfitting but limited capacity.
Positional Encoding	Fixed: More general. Learnable: Dataset-specific but potential overfitting.

5. Summary for Exam and Report

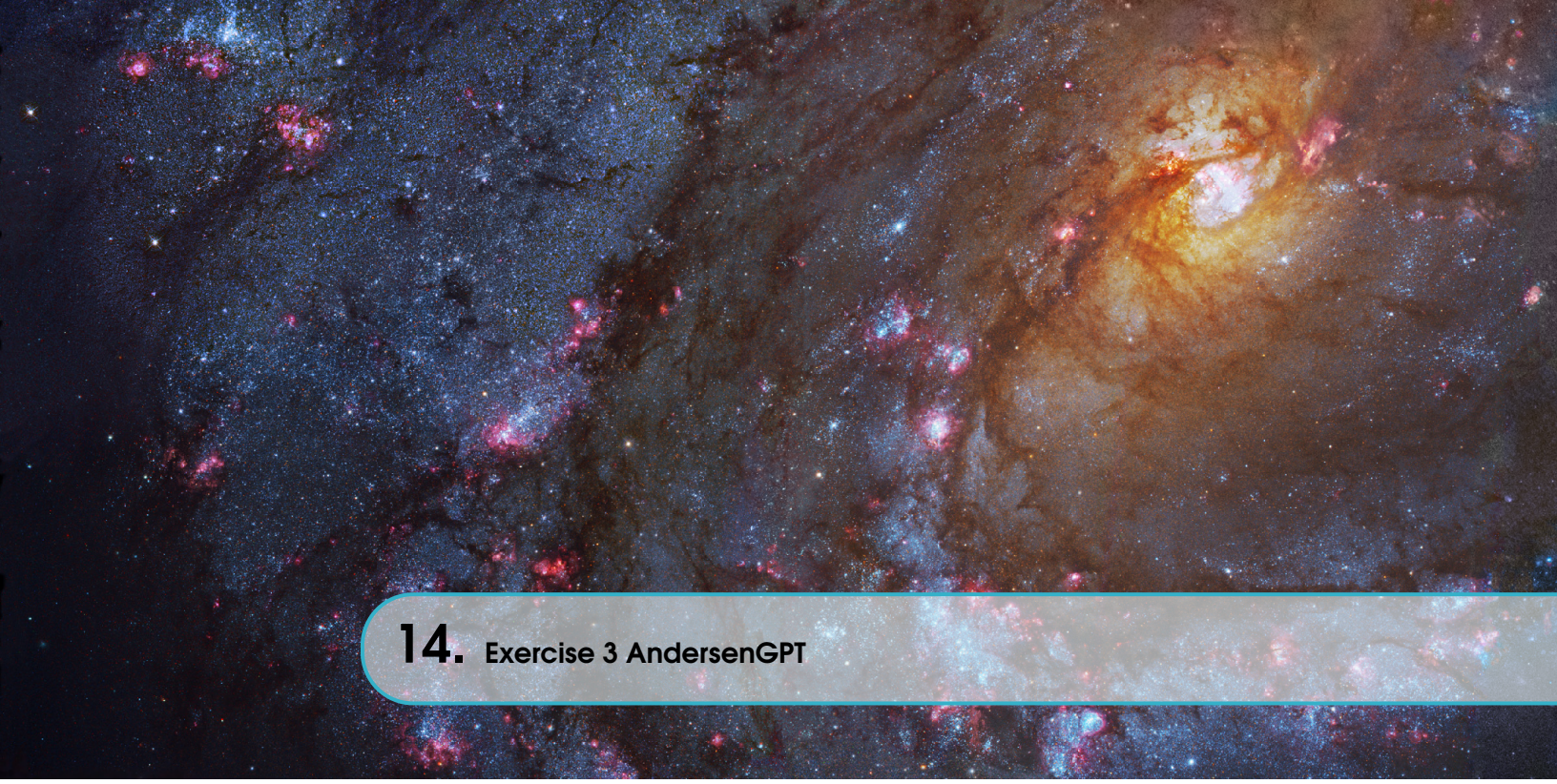
Key Learning Outcomes

- ViTs process images as **sequences of patches** rather than using convolutions
- ViTs capture **global context from the first layer**, unlike CNNs
- ViTs typically require **more data** than CNNs due to fewer inductive biases
- **Multi-head attention** allows different heads to focus on different image aspects
- **Positional encodings** are crucial for spatial understanding
- ViTs offer more **architectural flexibility** in scaling

Example Exam Questions

1. **Why does ViT require more data than a CNN?**
   Answer: ViTs lack the inductive biases of CNNs (locality, translation equivariance) that provide prior knowledge about images. Without these biases, ViTs must learn spatial relationships from scratch, requiring more examples.
2. **How do attention heads differ from convolutional filters?**
   Answer: Convolutional filters have fixed receptive fields and apply the same transformation across the image. Attention heads can dynamically focus on any part of the image regardless of distance, and their weights are input-dependent rather than fixed.

3. **How do positional embeddings impact classification performance?**
   Answer: Without positional embeddings, the model would treat patches as an unordered set, losing all spatial information. Positional embeddings help the model understand spatial arrangements and relationships between object parts — crucial for accurate classification.



14. Exercise 3 AndersenGPT

1. Core Transformer and GPT Concepts

Masked Attention Implementation

The first task in your exercise requires implementing the masked attention mechanism in the `MaskedAttention` class of `gpt.py`. This is a fundamental component of autoregressive language models like GPT.

What is Causal (Masked) Self-Attention?

Causal attention ensures that when predicting the next token, the model can only attend to previous tokens in the sequence (including the current one), not future ones. This is critical for autoregressive generation because:

- It prevents "cheating" during training by looking at future tokens
- It enables step-by-step text generation at inference time
- It maintains the probabilistic factorization:

$$p(x_1,\ldots,x_n) = p(x_1)\cdot p(x_2|x_1)\cdot\cdots\cdot p(x_n|x_1,\ldots,x_{n-1})$$

Differences from Bidirectional Attention

Unlike BERT (which uses bidirectional attention), GPT uses causal attention:

- **BERT:** Can see the entire context in both directions (useful for understanding)
- **GPT:** Can only see past and current tokens (necessary for generation)

This difference explains why BERT excels at understanding tasks (classification, NER) while GPT excels at generation tasks.

Transformer Decoder Architecture

The `EncoderBlock` in your code is actually functioning as a decoder block (despite its name). The structure features:

- **Layer Normalization** before attention and feed-forward (Pre-LN architecture)
- **Self-attention** with causal masking

- **Residual connections** after each sub-block
- **Feed-forward network** with GELU activation
- **Dropout** for regularization

The Pre-LN architecture (LayerNorm before attention) is important for stable training of deep transformer networks.

Positional Encoding

GPT needs to know the position of tokens in the sequence since self-attention is permutation-invariant. Your code implements two types:

- **Fixed sinusoidal encoding** (`PositionalEncoding`): Uses sine and cosine functions of different frequencies
- **Learnable positional embeddings** (`PositionalEmbedding`): Learns position representations during training

The exercise includes an opportunity to compare these approaches by setting the `pos_enc` parameter to either `"fixed"` or `"learnable"`.

2. Text Generation Strategies

For the autoregressive generation function in `test.py`, you'll implement:

Greedy Decoding

```
# If input_ids is too long, keep only the last MAX_SEQ_LEN tokens
if input_ids.size(1) > MAX_SEQ_LEN:
    input_ids = input_ids[:, -MAX_SEQ_LEN:]

# Forward pass to get logits for all tokens in the sequence
logits = model(input_ids)

# Get the logits for the last token only
next_token_logits = logits[:, -1, :]

# Greedy: choose the token with highest probability
next_token_id = next_token_logits.argmax(dim=-1, keepdim=True)

# Append predicted token to input_ids
input_ids = torch.cat([input_ids, next_token_id], dim=1)

# Stop if EOS token is generated
if next_token_id.item() == tokenizer.eos_token_id:
    break
```

Greedy decoding *always selects the highest probability token*, which leads to:

- More predictable, deterministic outputs
- Often repetitive and less creative text
- Tendency to get stuck in loops or repetitive patterns

Multinomial Sampling with Temperature

For the sampling strategy, you'll implement:

```
# Convert logits to probabilities with temperature
temperature = 0.8  # Adjust this value to control randomness
probabilities = F.softmax(next_token_logits / temperature, dim=-1)

# Sample from the probability distribution
next_token_id = torch.multinomial(probabilities, num_samples=1)
```

The temperature parameter controls the **"randomness"** of the sampling:

- $\mathbf{T < 1.0}$: More conservative, higher probabilities become even more likely
- $\mathbf{T = 1.0}$: Standard sampling from model distribution
- $\mathbf{T > 1.0}$: More exploratory, flattens the distribution for more variety

Higher temperatures produce more diverse but potentially less coherent text, while lower temperatures produce more focused but potentially repetitive text.

3. Experimental Design & Analysis

Training from Scratch vs. Fine-tuning

The exercise includes an option to train from scratch or fine-tune from a pretrained checkpoint:

```
if START_FROM_PRETRAINED_GPT2_CHECKPOINT:
    model.load_state_dict(torch.load("gpt2_pretrained.pt"))
```

Key considerations:

- **Fine-tuning** leverages knowledge from pretraining (faster convergence, better performance with less data)
- **Training from scratch** requires more data and computation but may better capture the specific style of Andersen

If you try both approaches, compare:

- Convergence speed (how quickly validation loss improves)
- Final model perplexity
- Qualitative assessment of generated text

Context Length Considerations

The exercise sets `MAX_SEQ_LEN = 1024`, but you could experiment with different values:

- **Shorter context**: Faster training and inference, but less coherence across longer stories
- **Longer context**: Better long-range dependencies, but more computationally expensive

For fairy tales, longer contexts may be valuable for maintaining narrative consistency and character development.

Sampling Strategies Comparison

In your report, analyze how different decoding strategies affect generated text:

- **Greedy**: Usually more coherent locally but can be repetitive
- **Sampling (low temp)**: More coherent with some variety
- **Sampling (high temp)**: More diverse but potentially less coherent

- **Beam search (optional)**: Tries to balance quality and diversity by exploring multiple paths

Include examples of text generated with different strategies to illustrate these differences.

4. Learning Objectives and Expected Outcomes

This exercise teaches you:

1. The internal mechanics of transformer-based language models
2. How to implement and train a GPT-style model from scratch
3. Different text generation strategies and their effects
4. The importance of design choices like context length and positional encoding

For your 4-page report, focus on:

- Clear explanation of your implementation choices
- Analysis of model training (loss curves, perplexity)
- Comparison of generation strategies (with examples)
- Discussion of how different parameters affect output quality

Oral Exam-Style Questions

Prepare for questions like:

1. Architectural Understanding

- Why does GPT use causal masking while BERT uses bidirectional attention?
- What would happen if you removed the causal mask in a GPT model?
- How do residual connections help in training deep transformer networks?

2. Generation Strategies

- Why is greedy decoding often suboptimal for creative text generation?
- What is the effect of temperature in multinomial sampling?
- What are the tradeoffs between beam search and multinomial sampling?

3. Training and Fine-tuning

- Why is fine-tuning from a pretrained model more data-efficient than training from scratch?
- How does the choice between fixed vs. learnable positional encodings affect model performance?
- What effect would increasing the context length have on training and output quality?

4. Implementation Details

- Why do we need to crop the input sequence to MAX_SEQ_LEN during generation?
- How does the model represent and learn word relationships?
- What role does the layer normalization placement (Pre-LN vs. Post-LN) play in training stability?

15. Exercise 2 Guided Diffusion Models

Understanding Guided Diffusion Models for Class-Conditional Image Generation

I'll explain the key concepts of guided diffusion models, focusing on **Classifier Guidance (CG)** and **Classifier-Free Guidance (CFG)** as applied to generating class-conditional images.

1. Fundamental Concepts of Diffusion Models

Diffusion models work in two phases:

- **Forward process:** Gradually adds noise to images until they become pure noise
- **Reverse process:** Learns to denoise images step by step to generate new samples

In standard diffusion models (DDPMs), the forward process is class-agnostic, meaning it doesn't consider class information when adding noise. However, the reverse process can be guided using class information to generate images from specific categories.

2. Two Approaches to Guided Diffusion

Classifier Guidance (CG)

Classifier Guidance uses a separate classifier that is trained to predict the class from noisy images at any timestep t. During sampling, the gradient of this classifier with respect to the noisy image is used to **"push"** the generation toward producing samples of a desired class.

The key mathematical insight is that we want to maximize:

$$\nabla_{x_t} \log p(y \mid x_t)$$

This gradient is added to the score function in DDPM to tilt the sampling path toward images of class y.

Implementation details:

1. The classifier must be trained to work on noisy images, not just clean ones

2. The timestep t must be provided as input to the classifier
3. During generation, the classifier's gradient is combined with the diffusion model's output

Classifier-Free Guidance (CFG)

Classifier-Free Guidance eliminates the need for a separate classifier. Instead, it trains a single conditional UNet to generate both conditional outputs (using labels) and unconditional outputs (by dropping labels at random during training).

Training process:

1. During training, labels are randomly dropped with some probability (usually 10–20%)
2. This teaches the model to produce both unconditional and conditional scores

Sampling formula:

At generation time, the outputs are combined using:

$$\varepsilon_{\text{guided}} = (1+w)\varepsilon_\theta(x_t, t, y) - w\varepsilon_\theta(x_t, t, \emptyset)$$

Where w is the guidance scale that controls the strength of conditioning.

Advantages of CFG:

- No external classifier needed
- Faster and more stable
- Enables semantic class mixing during generation by blending one-hot vectors

3. Key Implementation Steps

For Classifier Guidance:

1. Build a classifier that works on noisy images at any timestep
    - The classifier should take both noisy images and timesteps as input
    - Using the UNet encoder architecture is recommended
2. Train the classifier and save the weights in the appropriate location
3. Implement the gradient-based guidance during the sampling process

For Classifier-Free Guidance:

1. Modify the UNet to accept one-hot encoded class labels as input
    - Add a layer that projects the labels to the time embedding dimension
2. Combine the label embedding with the time embedding
3. Implement the training algorithm that randomly discards labels:
    - With probability p_{uncond}, replace the label with null ($\emptyset$)
    - Train the model to predict the noise based on the noisy image, timestep, and (possibly dropped) label
4. Implement the sampling procedure that combines conditional and unconditional predictions

4. Quantitative Evaluation with FID

Fréchet Inception Distance (FID) is a metric used to measure the similarity between generated and real images in the feature space of an Inception model.

It compares the mean and covariance of real vs fake distributions:

$$\text{FID} = \|\mu_r - \mu_g\|^2 + \text{Tr}(\Sigma_r + \Sigma_g - 2(\Sigma_r\Sigma_g)^{1/2})$$

Where μ_r, μ_g are the mean feature vectors and Σ_r, Σ_g are the covariance matrices for real and generated images, respectively.

Best practices for FID evaluation:

- Evaluate FID on the same dataset split (should be near 0)
- Test with pure noise (should give very high FID)
- Compare CG vs CFG scores to evaluate generation quality vs controllability

5. Comparison of Approaches

Aspect	**Classifier Guidance**	**Classifier-Free Guidance**
Architecture	Separate classifier + DDPM	Single conditional model
Training	Train two models separately	Train one model with label dropout
Sampling	Use classifier gradients	Combine conditional/unconditional outputs
Advantages	Powerful control	Simpler, faster, enables class mixing
Disadvantages	Extra compute, needs accurate classifier	May have less precise control

6. Class Mixing (Optional)

One interesting capability of Classifier-Free Guidance is the ability to mix classes during generation. Since the model takes one-hot encoded vectors as input, you can create blended vectors (like $0.7\cdot$ [`human`] $+0.3\cdot$ [`food`]) to generate images with mixed characteristics.

Conclusion

Both Classifier Guidance and Classifier-Free Guidance offer effective ways to control the output of diffusion models. CG uses an external classifier's gradients, while CFG incorporates conditioning directly into the diffusion model. The FID metric allows for quantitative comparison between these approaches, helping to determine which performs better for specific generation tasks.

Understanding these guidance techniques is crucial for creating controllable generative models that can produce high-quality images from specific categories or with desired attributes.

16. Exercise 3: Textual Inversion

Core Concept of Textual Inversion

Textual Inversion is a technique that allows you to teach a pre-trained text-to-image model (like Stable Diffusion) new visual concepts using just a few example images. The key insight is that you can represent a new visual concept by optimizing a single embedding vector in the text encoder's embedding space, while keeping the rest of the model frozen.

In Simpler Terms

- You want the model to learn a new concept (like a specific character, style, or object)
- Instead of fine-tuning the whole model, you only train a new "word" (embedding vector)
- This new "word" can then be used in prompts just like any other word

How It Works

1. **Adding a New Token:** You add a special placeholder token (like `<my-concept>`) to the model's tokenizer.
2. **Initializing the Embedding:** You initialize this token's embedding, either:
   - Randomly (from a normal distribution)
   - By copying from a semantically similar token (e.g., "toy" for a toy concept)
3. **Freezing the Model:** You freeze all parameters of the model:
   - The CLIP text encoder
   - The U-Net denoiser
   - The VAE encoder/decoder
4. **Training Loop:** During training, you:
   - Process your concept images through the VAE to get latents
   - Add noise to these latents
   - Create text prompts with your placeholder token
   - Predict the noise using the U-Net

- Calculate the loss between predicted and actual noise
- Update only the embedding of your placeholder token

5. **Using the Trained Embedding:** After training, your token's embedding has been optimized to represent your visual concept. You can use it in prompts like `"A <my-concept> on a beach"` to create new compositions.

Why It Works

This works because diffusion models are conditioned on text embeddings from a text encoder (like CLIP). The text encoder maps tokens to a semantic latent space where similar concepts are nearby. By optimizing a new embedding in this space, you're essentially finding the "coordinates" that best represent your visual concept.

The beauty of this approach is that it leverages the composition capabilities of the text encoder. Since your new token lives in the same embedding space as other words, you can compose it with other concepts naturally.

Key Technical Details

For the Training Loop Implementation

You need to:

1. Get a batch of images and convert them to latent space using the VAE
2. Sample noise and timesteps
3. Add noise to the latents
4. Get text embeddings for prompts containing your placeholder token
5. Predict the noise with the U-Net
6. Calculate the loss between predicted and actual noise
7. Update only the embedding of your placeholder token
8. Apply gradient clipping to stabilize training

For Concept Generation

Once trained:

1. Load the model with your trained embedding
2. Create prompts that include your placeholder token in different contexts
3. Generate images with different guidance scales (higher for more faithful reproduction)
4. Evaluate the quality using metrics like FID and CLIP score

Important Concepts to Understand

1. **Parameter Efficiency:** You're only training a single embedding vector (~768 dimensions) instead of millions or billions of parameters.
2. **Compositionality:** The new concept inherits the compositional properties of language, so it can be combined with other concepts.
3. **Regularization:** Techniques like gradient clipping help prevent the embedding from drifting too far in the embedding space.
4. **Data Efficiency:** Textual Inversion works with very few example images (3–5 is often enough).

5. **Cross-Attention Mechanism:** How the text embeddings condition the diffusion process through cross-attention in the U-Net.

17. Stable Diffusion vs Stable Diffusion XL

Stable Diffusion

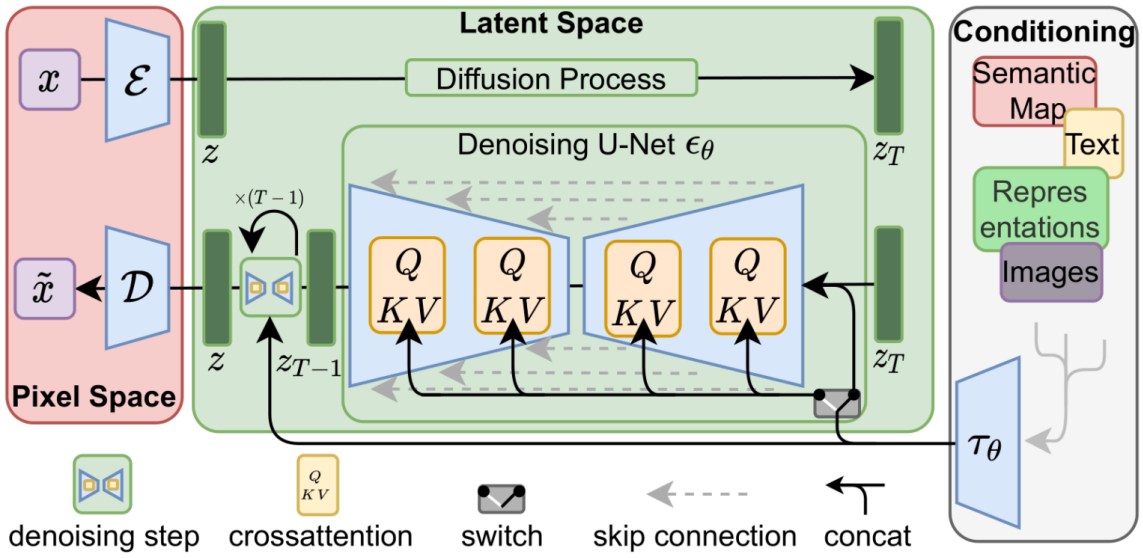


Figure 17.1: Architectural diagram of a conditional diffusion model. The system operates between Pixel Space (left) and Latent Space (center), with various conditioning inputs (right). The process begins with an encoder ($\mathcal{E}$) mapping input images (x) to latent representations (z), followed by a forward diffusion process reaching noise level z_T. The denoising phase employs a U-Net (ϵ_θ) with cross-attention mechanisms (Q,K,V) that incorporate conditioning signals (τ_θ) from text, semantic maps, and other representations. The iterative denoising process transitions from z_T to z_{T-1} and continues until reaching z, after which the decoder ($\mathcal{D}$) reconstructs the output image ($\tilde{x}$). Skip connections (dashed arrows) facilitate information flow across the U-Net, while the concatenation operations (black arrows with T-junctions) integrate features at multiple scales.

1. Denoising Diffusion Probabilistic Models (DDPM)

- **Generative process:**
  - Forward: add Gaussian noise in T small steps to a clean image x_0, producing noisy x_t.
  - Reverse: learn a denoiser $\varepsilon_\theta(x_t, t)$ to recover x_{t-1} from x_t.
  - Objective: minimize

$$\mathscr{L}_{\text{DDPM}} = \mathbb{E}_{x,\varepsilon\sim\mathcal{N}(0,1),t}\left[\|\varepsilon - \varepsilon_\theta(x_t,t)\|^2\right].$$

- **Drawbacks:**
  - Operates in high-dimensional *pixel* space, so each sampling step requires a full UNet pass on the full image.
  - Typically uses hundreds to thousands of timesteps: inference is slow (e.g. 50 K samples take 5 days on A100) and training can cost ∼150–1000 V100-GPU-days.
  - Requires large datasets to learn pixel-space noise-removal.

2. Stable Diffusion (Latent Diffusion Model, LDM)

- **Key idea:** Perform diffusion not in pixels but in a learned *latent* space of a powerful autoencoder.
- **Two-stage training:**
  1. *Perceptual compression stage:* Train an autoencoder (E, D) (with a small KL- or VQ-regularization) to map $x \in \mathbb{R}^{H\times W\times 3}$ to $z \in \mathbb{R}^{h\times w\times c}$, where $h = H/f$ (e.g. $f = 4$).
  2. *Latent diffusion stage:* Train a time-conditional UNet $\varepsilon_\theta(z_t, t)$ to denoise z_t back to z_{t-1}, minimizing

$$\mathscr{L}_{\text{LDM}} = \mathbb{E}_{z,\varepsilon,t}\left[\|\varepsilon - \varepsilon_\theta(z_t,t)\|^2\right]$$

     where $z_0 = E(x)$ and z_t follows the fixed forward diffusion in latent space.
- **Efficiency gains:**
  - Latent dimensionality is much lower ($h\times w \ll H\times W$), so each denoising step is far cheaper.
  - Overall training and sampling speed can improve by $\sim 3\times$ or more compared to a pixel DDPM .
- **Conditional generation:**
  - Incorporate arbitrary conditioning (e.g. text) via a domain encoder $\tau_\phi(y)$ plus cross-attention layers in the UNet backbone
  - At inference: use classifier-free guidance to trade off fidelity vs. adherence to y.
- **Decoding:** After denoising to $\hat{z}_0$, reconstruct $\hat{x} = D(\hat{z}_0)$.

3. Main Differences DDPM vs. Stable Diffusion

- **Operational domain:** DDPM acts on full RGB pixels; SD acts on compressed latents.
- **Compute & memory:** DDPM: high FLOPs/GPU memory per step (full image). SD: lower cost per step (small latent), faster sampling training.
- **Model structure:** DDPM = single UNet over pixels. SD = (pretrained autoencoder) + latent UNet + optional text encoder + cross-attention.
- **Flexibility:** SD reuses one autoencoder for many tasks, and its cross-attention easily supports multi-modal conditioning (text, layouts, masks), whereas DDPMs typically use simple concatenation or class conditioning.

- **Quality vs. efficiency trade-off:** SD finds a near-optimal balance between compression and detail preservation by choosing f (often 4–8) .

1. Comparing Stable Diffusion (SD) vs. SDXL

- **Overall pipeline:** Both SD and SDXL are Latent Diffusion Models (LDMs) that operate in a compressed latent space via a pretrained VAE (E, D), apply a time-conditional UNet for denoising, then decode back to pixel space.
- **UNet backbone size:**
  - *SD (v1.4/2.1)*: $\sim$1 B parameters in the U-Net denoiser.
  - *SDXL 1.0*: a *three times larger* UNet backbone (3.5 B parameters), achieved by adding more attention blocks and increasing the hidden dimensionality .
- **Text conditioning:**
  - *SD*: single CLIP-style text encoder, cross-attention at each UNet layer.
  - *SDXL*: *two* text encoders (e.g. OpenCLIP-ViT/G CLIP-ViT/L), allowing dual-prompt or more expressive guidance; cross-attention uses the concatenated context from both encoders.
- **Aspect ratios & resolutions:**
  - *SD*: trained primarily at square 512×512 (later 768×768) resolutions.
  - *SDXL*: trained on *multiple aspect ratios* (e.g. 1:1, 3:2, 2:3, 1:2, 2:1) to retain fidelity across portrait/landscape formats .
- **Refinement stage:**
  - *SD*: single diffusion denoiser.
  - *SDXL*: includes an *optional refiner* UNet, applied post-hoc in an img2img fashion to sharpen and add fine details to the base SDXL .
- **Sampling efficiency:**
  - Larger models slow down per-step inference, but SDXL's improved capacity often requires fewer steps for equivalent or better quality.
  - A distilled "XL Turbo" variant further reduces the number of denoising steps needed.

2. Explanation of the Stable Diffusion Pipeline Diagram

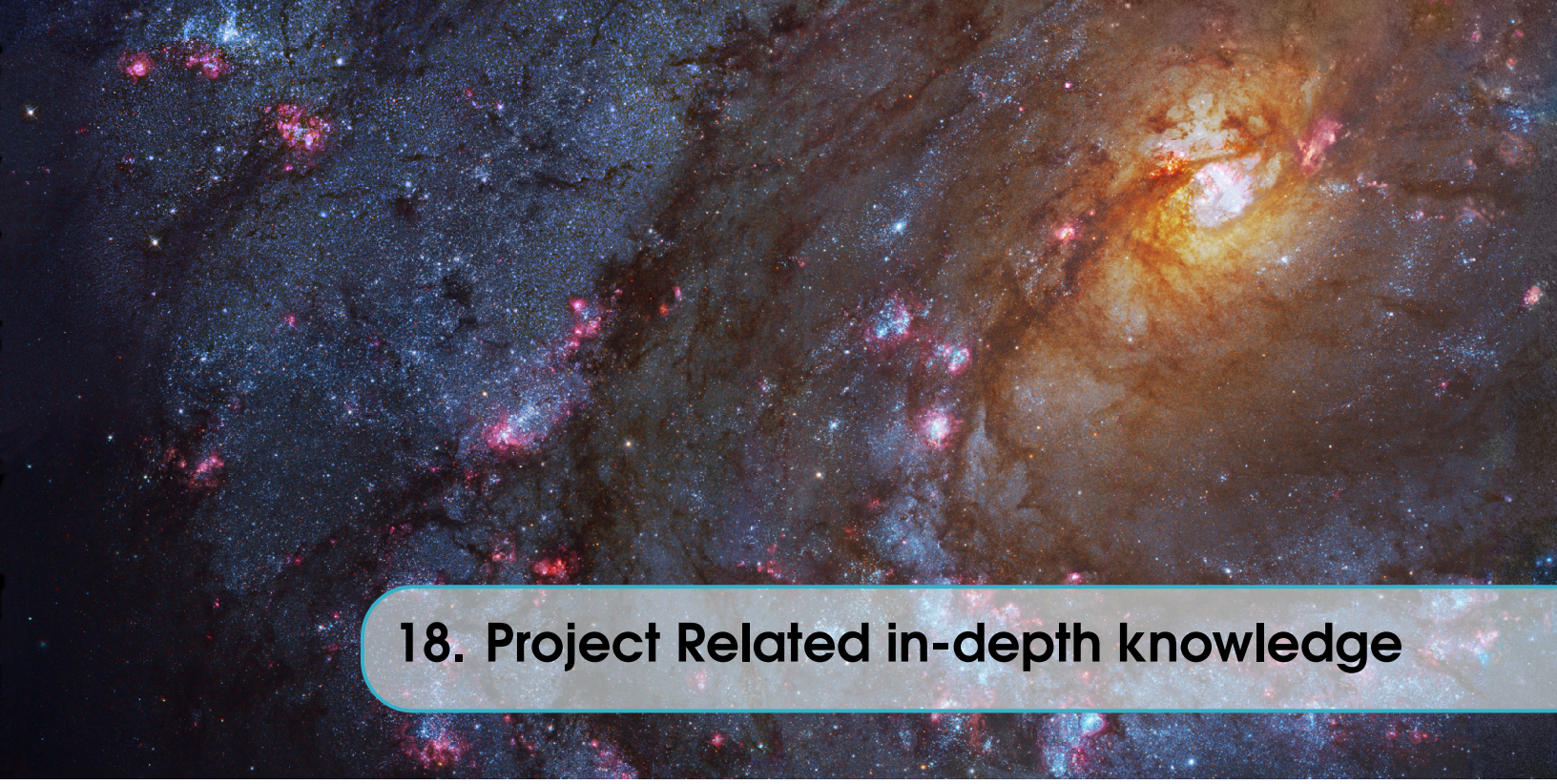
A canonical SD diagram (cf. Fig. **??**) breaks down into:

1. **VAE Stage:**
   - *Encoder E:* maps pixel image $x \in \mathbb{R}^{H\times W\times 3}$ to latent $z = E(x) \in \mathbb{R}^{h\times w\times c}$.
   - *Decoder D:* reconstructs $\tilde{x} = D(z)$ for training the VAE; frozen at diffusion-training time.
2. **Forward diffusion (latent):**

$$z_t \;=\; \sqrt{\bar{\alpha}_t}\, z_{t-1} \;+\; \sqrt{1-\bar{\alpha}_t}\, \varepsilon, \quad \varepsilon \sim \mathcal{N}(0, I),$$

   gradually corrupting z with Gaussian noise over T timesteps.
3. **Denoising UNet $\varepsilon_\theta(z_t, t, y)$:**
   - *Time embedding* encodes t.
   - *Cross-attention* layers inject conditioning (text embeddings or other) into intermediate UNet blocks.
   - *Skip connections* shuttle high-resolution details from encoder to decoder paths.
   - Predicts noise estimate $\hat{\varepsilon}$ to recover z_{t-1} from z_t.

4. **Reverse diffusion sampling:** Iteratively apply the learned denoiser, optionally with classifier-free guidance, to obtain $\hat{z}_0$.
5. **Reconstruction:** Decode $\hat{z}_0$ back to pixel space: $\hat{x} = D(\hat{z}_0)$, yielding the final generated image.



18. Project Related in-depth knowledge

18.1 Problem

Text

18.2 Solution

18.3 Data

18.4 Data preprocessing

18.5 Models

18.5.1 Vision Language Models

18.5.2 Diffusion Models

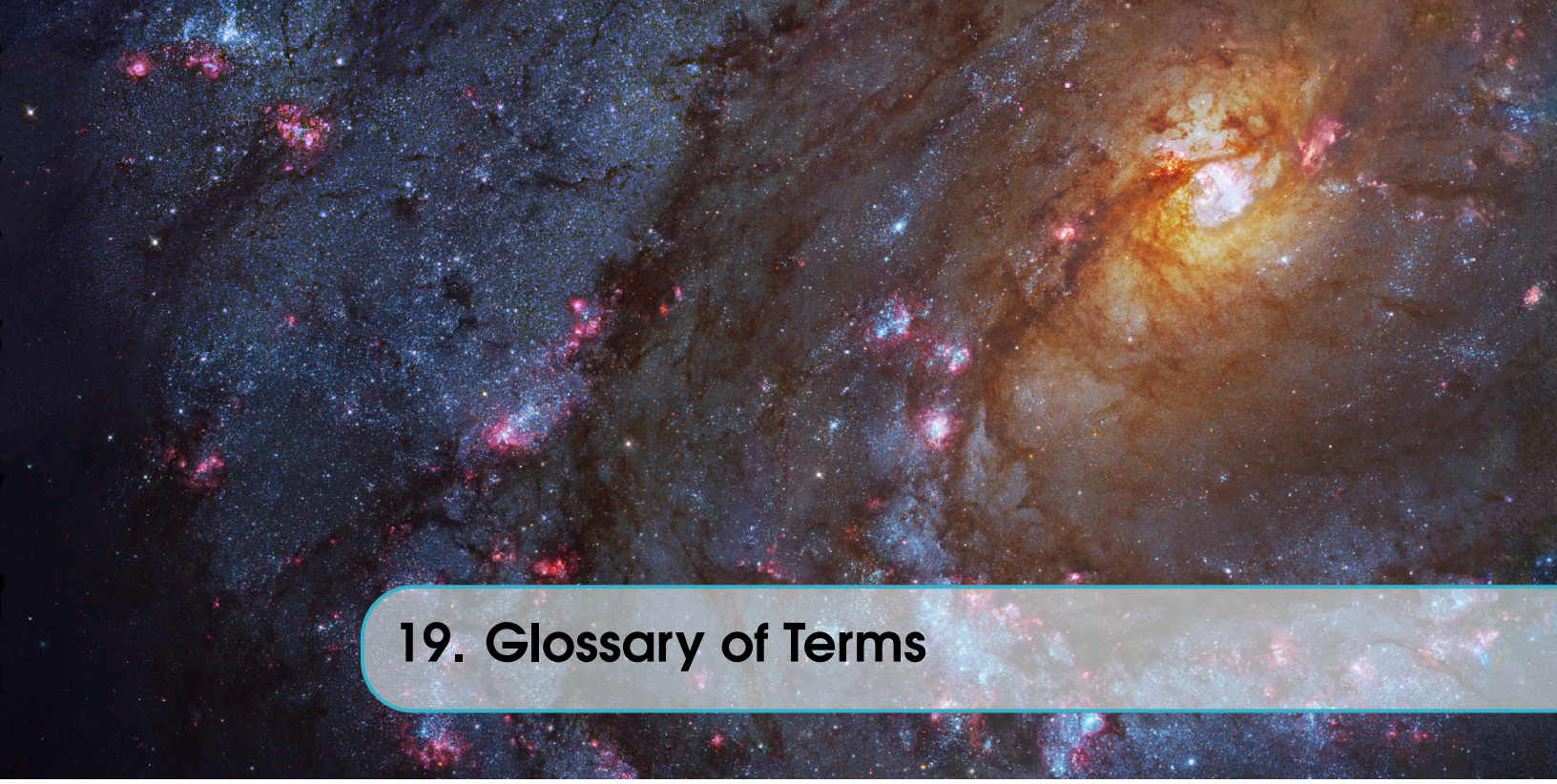
18.5.3 Vision Transformer

18.6 Training

18.6.1 Parameters

18.6.2 Loss Functions

18.7 Results



19. Glossary of Terms

19.1 Explainability and Fairness

Explainable AI (XAI): Methods and techniques that allow human users to understand, trust, and effectively manage AI systems. In computer vision, it often aims to explain model predictions.

Counterfactual XAI: An explanation that answers "How should the input change to obtain a different classification?" It involves identifying minimal changes to an input that alter the model's prediction.

Algorithmic Fairness: The principle and methods aimed at reducing unwanted bias in algorithmic decision-making, ensuring outcomes are equitable across different groups.

Independence (Algorithmic Fairness): A notion of fairness where the prediction (or acceptance rate) is independent of group membership.

Separation (Algorithmic Fairness): A notion of fairness requiring equal rates of True Positives, False Positives, True Negatives, and False Negatives across different groups.

Sufficiency (Algorithmic Fairness): A notion of fairness requiring equal positive and negative predictive values across different groups, meaning given a prediction, the probability of it being true is the same across groups.

19.2 Transformers and Language Models

Transformer: A neural network architecture that relies heavily on self-attention mechanisms to process sequential data, initially developed for natural language processing but adapted for computer vision (Vision Transformers).

Transformer Encoder: Part of the original Transformer architecture responsible for processing the input sequence in parallel and generating contextualized representations.

Transformer Decoder: Part of the original Transformer architecture responsible for generating the output sequence autoregressively, often utilizing masked attention and cross-attention.

Self-Attention: A mechanism within Transformers that allows each element in a sequence to weigh the importance of all other elements in the same sequence when computing its representation.

Multi-Head Attention: An extension of self-attention where the attention mechanism is applied multiple times in parallel with different learned linear projections of the input.

Positional Encoding: A method used in Transformers to inject information about the position of elements in a sequence, as self-attention is permutation-equivariant.

Masked Attention: An attention mechanism used in Transformer Decoders that prevents attention to subsequent positions in the sequence, ensuring autoregressive generation.

Cross-Attention: An attention mechanism used in Transformer Decoders (in Encoder-Decoder models) where the queries come from the decoder's sequence and the keys and values come from the encoder's output sequence.

BERT (Bidirectional Encoder Representations from Transformers): An encoder-only LLM trained using Masked Language Modeling, focusing on understanding and classifying text.

GPT (Generative Pre-trained Transformer): A decoder-only LLM trained using Autoregressive Language Modeling, focusing on generating text.

Large Language Models (LLMs): Language models with a very large number of parameters, trained on vast amounts of text data, capable of performing various NLP tasks including text generation.

Tokenization: The process of converting text into discrete units (tokens) that can be processed by a language model.

Decoding Strategies: Algorithms used to select the next token during the generation process in autoregressive models, examples include Greedy, Beam Search, Multinomial Sampling, Top-K, and Top-P sampling.

19.3 Generative Models and Diffusion

Generative AI: AI models capable of creating new data, such as images, text, or audio, that are similar to the data they were trained on.

Generative Model: A model that describes a probability distribution and can sample from it to generate new data points.

Density Modelling: The task of learning the underlying probability distribution of a dataset. Generative models often perform this implicitly or explicitly.

Diffusion Models: A class of generative models that work by gradually adding noise to data (forward process) and then learning to reverse this process to generate new data from noise (reverse process).

Forward Diffusion Process: The process of gradually adding noise to an image over time steps, defined as a Markov chain.

Reverse Diffusion Process: The process of gradually removing noise from a noisy image over time steps to generate a clean image, which is learned by the diffusion model.

Variance Schedule (β_t): A sequence of hyperparameters that determine the scale of noise added at each step in the forward diffusion process.

Denoising: The task of removing noise from an image; this is what the neural network within a diffusion model learns to do at each step of the reverse process.

U-Net: A convolutional neural network architecture commonly used for image-to-image tasks like segmentation and denoising, often employed as the backbone for diffusion models.

Guidance: Techniques used to steer the generative process of Diffusion Models towards desired attributes, such as generating images belonging to a specific class or matching a text prompt.

Classifier Guidance: A guidance technique for Diffusion Models that uses the gradients of a

separate classifier with respect to the image to steer the generation towards a target class.

Classifier-Free Guidance: A guidance technique that trains a single Diffusion Model capable of both unconditional and conditional generation, using a weighted combination of the scores from both to achieve guidance.

Counterfactual Explanations (for image classifiers): Explanations that show the minimal changes required to an image input to change its classification by a model.

Universal Guidance: A guidance framework for Diffusion Models that can incorporate guidance from various modalities (e.g., text prompts, segmentation masks, bounding boxes) using a shared loss function.

Latent Diffusion Models: Diffusion Models that operate in a lower-dimensional latent space learned by an autoencoder, rather than directly in the pixel space, making them more efficient for high-resolution images.

Stable Diffusion: A popular latent diffusion model capable of high-quality text-to-image generation and other image manipulation tasks.

19.4 Self-Supervised and Multimodal Learning

Self-Supervised Learning (SSL): A machine learning paradigm where a model learns from unlabelled data by training on "pretext tasks" where the labels are derived automatically from the data itself.

Pretext Tasks: Tasks used in self-supervised learning to train a model without requiring human annotations, such as predicting missing parts of an image or learning spatial relationships.

Contrastive Learning: An SSL approach that learns representations by pulling together embeddings of augmented versions of the same image (positives) and pushing apart embeddings of different images (negatives).

MAE (Masked Autoencoders): An SSL approach that trains a Vision Transformer to reconstruct missing patches of an image.

DINO (Self-supervised Vision Transformers): An SSL approach that trains Vision Transformers by self-distillation, where a "student" network learns from a "teacher" network of the same architecture.

Multimodal Learning: Machine learning that processes and relates information from multiple modalities, such as combining visual and textual data.

CLIP (Contrastive Language–Image Pre-training): A multimodal model trained to connect images and text using a contrastive objective, enabling zero-shot image classification and text-to-image generation.

Visual Question Answering (VQA): A multimodal task where a model is given an image and a natural language question about the image and must provide a natural language answer.

Visual Reasoning: Multimodal tasks requiring deeper understanding and inference about the spatial and semantic relationships between objects in an image based on text queries.

Image-Text Contrastive (ITC) Loss: A loss function used in multimodal training (like BLIP) to align representations from different modalities (image and text) in a shared embedding space.

Image-Text Matching (ITM) Loss: A loss function used in multimodal training (like BLIP) to determine if a given image-text pair is a positive match or a negative mismatch.

Language Modeling (LM) Loss: A loss function used in training language models or the text generation component of multimodal models (like BLIP) to predict the next token in a sequence.

Grounding DINO: An open-set object detection model that can detect objects based on arbitrary text prompts, combining object detection and language grounding.

Textual Inversion: A technique for personalizing generative models (like Stable Diffusion) by learning a new "word" (a text embedding) in the model's vocabulary to represent a specific concept or object from a few example images.

Zero-shot Classification: The ability of a model to classify images into categories it has not seen during training, often enabled by multimodal models like CLIP by relating images to text descriptions of categories.

19.5 Miscellaneous

Autoregressive: A property of models that generate sequences one element at a time, conditioning on the previously generated elements.

Markov Chain: A stochastic process where the future state depends only on the current state, not on the sequence of events that preceded it. Used to define the forward process in diffusion models.

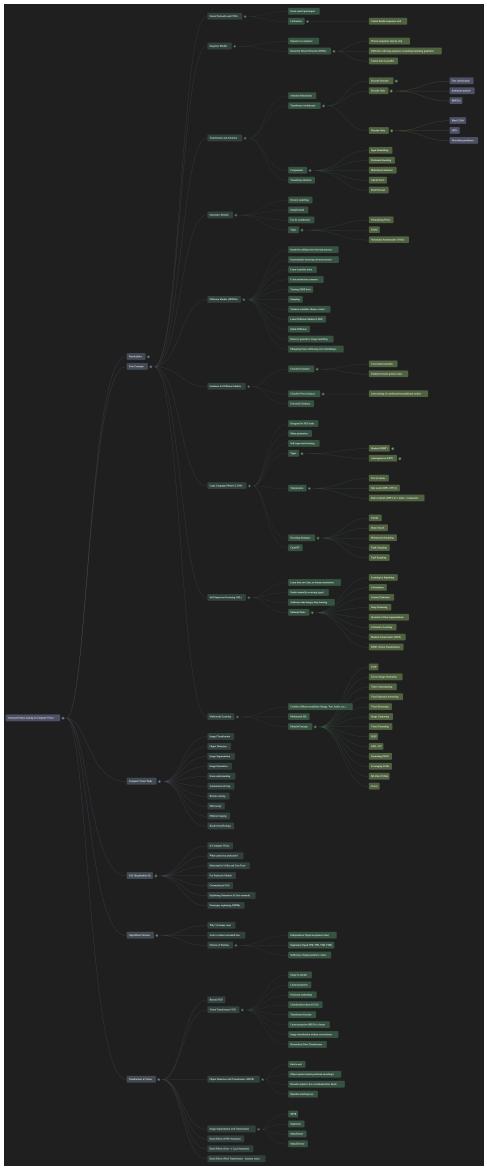


Figure 19.1: Mind map of the whole course

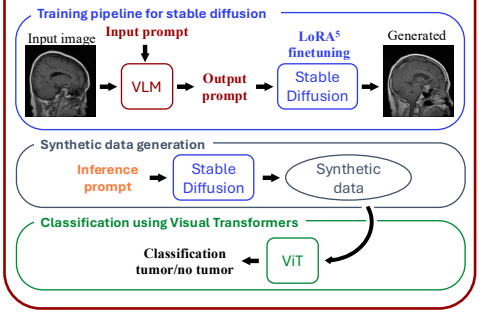
DTU Compute
Department of Applied Mathematics and Computer Science

Generative Modeling of High Fidelity Brain Tumor MRI Images Using Vision Language & Stable Diffusion Models

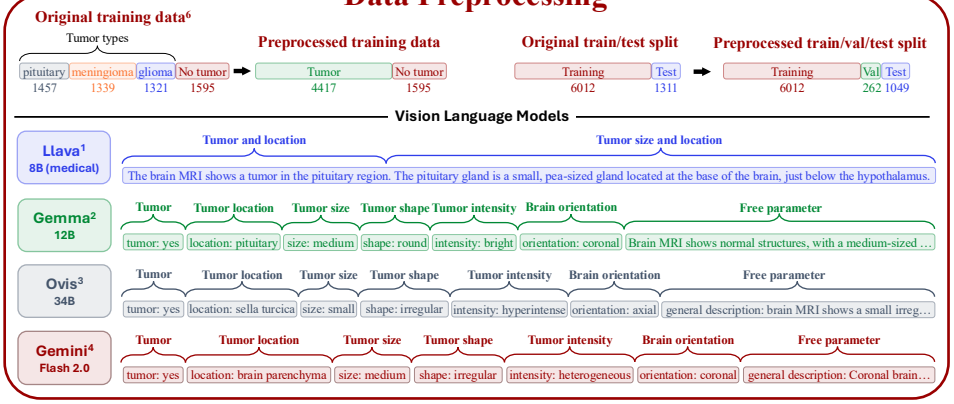
Jone Steinhoff (s243867), Lukas Rasocha (s233498), Mads Prip (s240577) & Petr Boska Nylander (s240466)

Introduction

The project aims to generate synthetic MRI tumor images with controllable tumor locations via Stable Diffusion prompts, and evaluate quality using FID, KID, and a classification task.



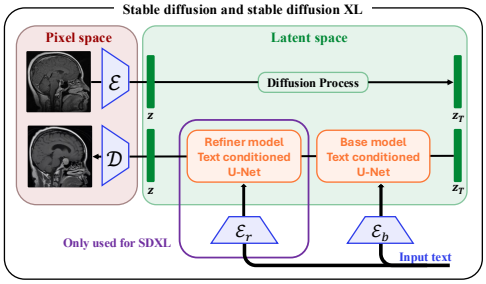
Data Preprocessing

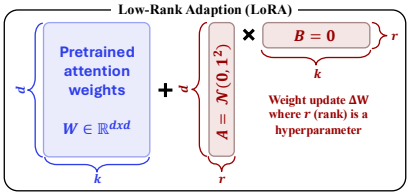


Models

Stable diffusion XL hyperparameters

- Rank **128/248**
- Resolution **512**
- Batch Size **2**
- Accumulation Steps **8**
- Learning Rate **1e-4**
- Gradient Checkpointing **True**
- Learning rate Scheduler **Cosine**
- SNR Gamma **5**
- Adam Weight Decay **0.01**
- Learning Rate Warmup Steps **1000**





Stable diffusion hyperparameters

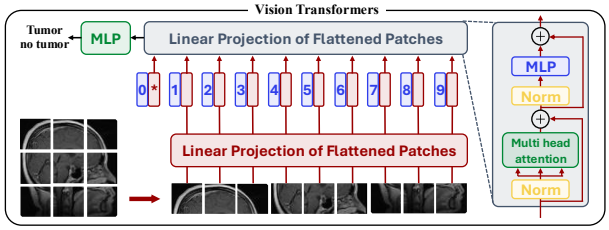
- Rank **128/248**
- Resolution **512**
- Batch Size **2**
- Accumulation Steps **8**
- Learning Rate **1e-4**
- Gradient Checkpointing **True**
- Learning rate Scheduler **Cosine**
- SNR Gamma **5**
- Adam Weight Decay **0.01**
- Learning Rate Warmup Steps **500**

Stable diffusion inference

- Number of Denoising Steps **50**

Vision Transformers hyperparameters

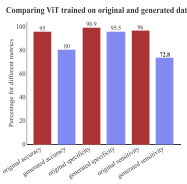
- Batch Size **16**
- Image Size **512**
- Patch Size **16**
- Channels **3**
- Embedding Dimension **128**
- Number of Heads **4**
- Number of Layers **1**
- Number of Classes **2**
- Positional Encoding **Learnable**
- Pooling Method **CLS**
- Dropout Rate **0.3**
- Fully Connected Dimension **512**
- Number of Epochs **40**
- Learning Rate **1e-4**
- Warmup Steps **625**
- Weight Decay **1e-3**
- Gradient Clipping **1.0**

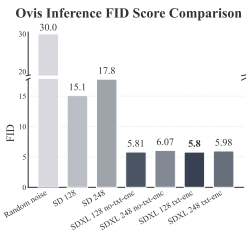


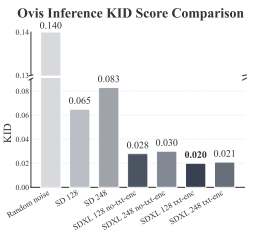
Results

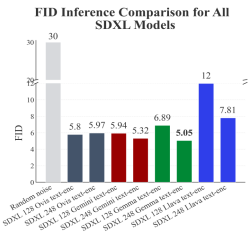
KID and FID scores show, that SDXL fine-tuned on Ovis with a trainable text encoder performs the best. Llava, Gemma and Gemini are used in the same setup to enable comparison.

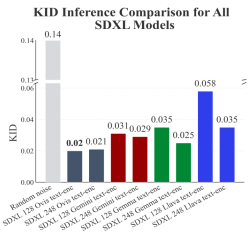
The best-performing configurations are based on Ovis (for KID) and Gemma (for FID), both incorporating a trainable text encoder and LoRA ranks of 128 and 248, respectively. Synthetic data generated with Ovis is used to classify tumor vs. non-tumor cases, in comparison to the original dataset. Accuracy, specificity and sensitivity are being compared.



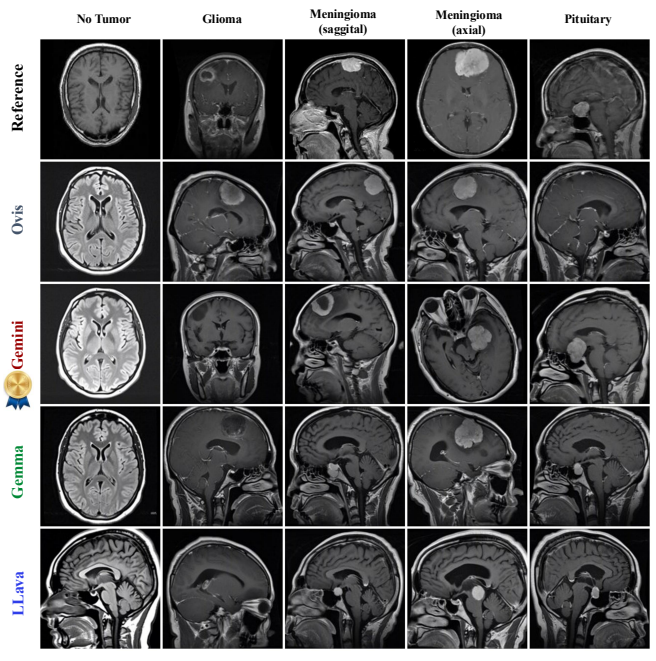








Model Acc.	Tumor	Loc.	Size	Shape	Intensity	Orient.
Ovis	1	0.66	0.65	0.5	**0.8**	0.35
Llava	1	0.45	0.05	0	0.05	0
Gemini	1	**0.82**	0.65	**0.7**	**0.8**	**1**
Gemma	1	0.47	**0.66**	0.5	0.64	0.8



Conclusion

- Combining automatic data labeling with VLMs with fine-tuning large diffusion models shows potential in domains with limited data availability.
- The choice of VLM impacts the quality of generated images. The best VLMs in our set-up were **Ovis 34B (KID)** and **Gemma 13B (FID).**
- SDXL consistently showed higher generative quality compared to SD-v1-5.
- Using detailed and structured medical prompts to control the generation of MRI scans shows potential, where **Gemini** prompts were followed the best.

References

1. https://github.com/microsoft/LLaVA-Med
2. https://huggingface.co/google/gemma-3-12b-it
3. https://huggingface.co/AIDC-AI/Ovis2-34B
4. https://ai.google.dev/gemini-api/docs/models
5. https://doi.org/10.48550/arXiv.2106.09685
6. https://doi.org/10.34740/kaggle/dsv/2645886