LEARNING FROM UNLABELED DATA BY EXPLOITING STRUCTURE WITHIN THE DATA ITSELF.

Self-Supervised Learning (SSL) Fundamentals

Recap of Key Concepts

Reminders

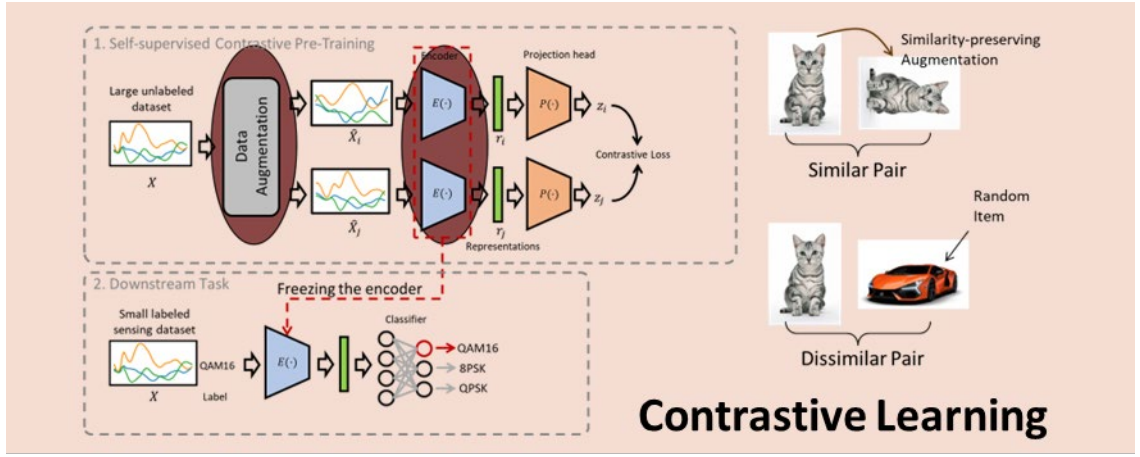
- Projects should be 2-3 people. If your project partners dropped the class and you are alone, please let me know. I will assign you a new group.
- Book (at least) a bi-weekly project zoom meeting with me asap. A list of bi-weekly slots starting next week or the week after is available for booking at the link below. Book one for a bi-weekly meeting or two in consecutive weeks for a weekly meeting. Please indicate your group number when booking.
  - https://doodle.com/meeting/participate/id/bqplRv7a
- Your project title and abstract are due Feb 10th. (Feel free to discuss the topic with me ahead of time on your group channel on Piazza, especially if you anticipate needing hardware.)
  - I will read your abstract and give you feedback on the project
  - You will generate a 2-page project description/plan by Feb 26th. It should include (i) the key topic (what are you going to do), (ii) what's innovative/different about it compared to the state of the art, (iii) why should people care, and (iv) an approximate execution timeline.
- Student-led talks:
  - There are 12 topics to choose from (see schedule). By Thursday 2/5, please bid on topics you'd like to cover by sending a post on your group channel on Piazza identifying your first, second, and third topic choices.
  - I will assign 1-2 topics per group. If your group is assigned 1 topic (those are mostly before Spring break), please expect to summarize 6 key papers in your talk. If your group is assigned 2 topics (those are mostly after Spring break), please expect to summarize 3 papers per topic.
  - Student-led talk assignments will be announced this Friday (together with recommended papers per topic).

What is Self-Supervised Learning?

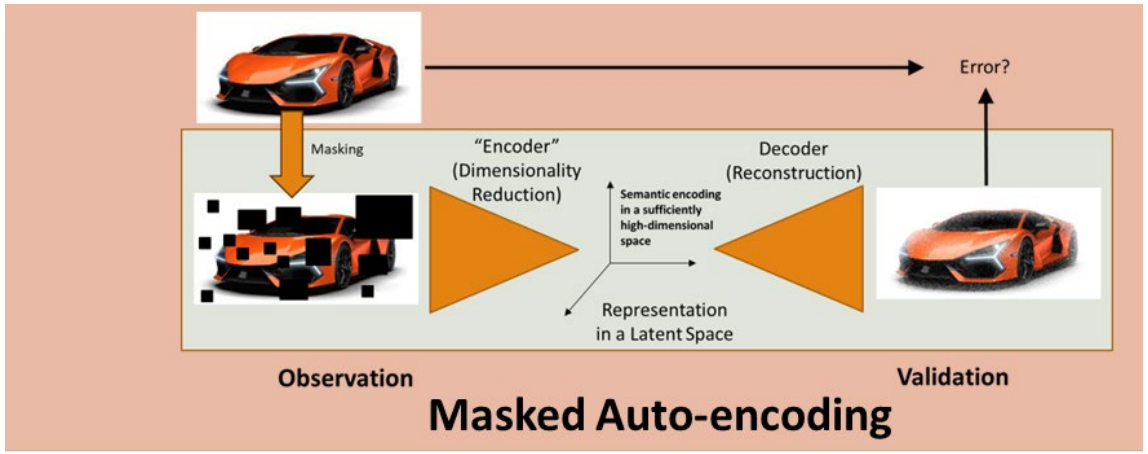
- No human-labeled data
- "Supervision" is generated automatically
- Goal: learn useful data representations

Components of Self-Supervised Learning

- **What are the key components of self-supervised learning (SSL)?**



SSL Example 1



SSL Example 2

Components of Self-Supervised Learning

Part I: The tokenizer

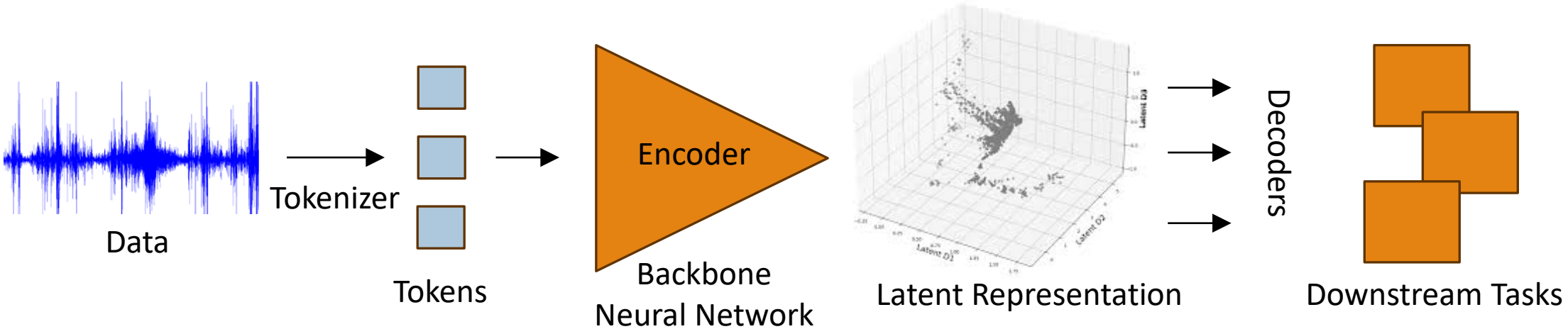
- Breaks the input stream into pieces to be individually encoded.

Part II: The backbone neural network model (or encoder)

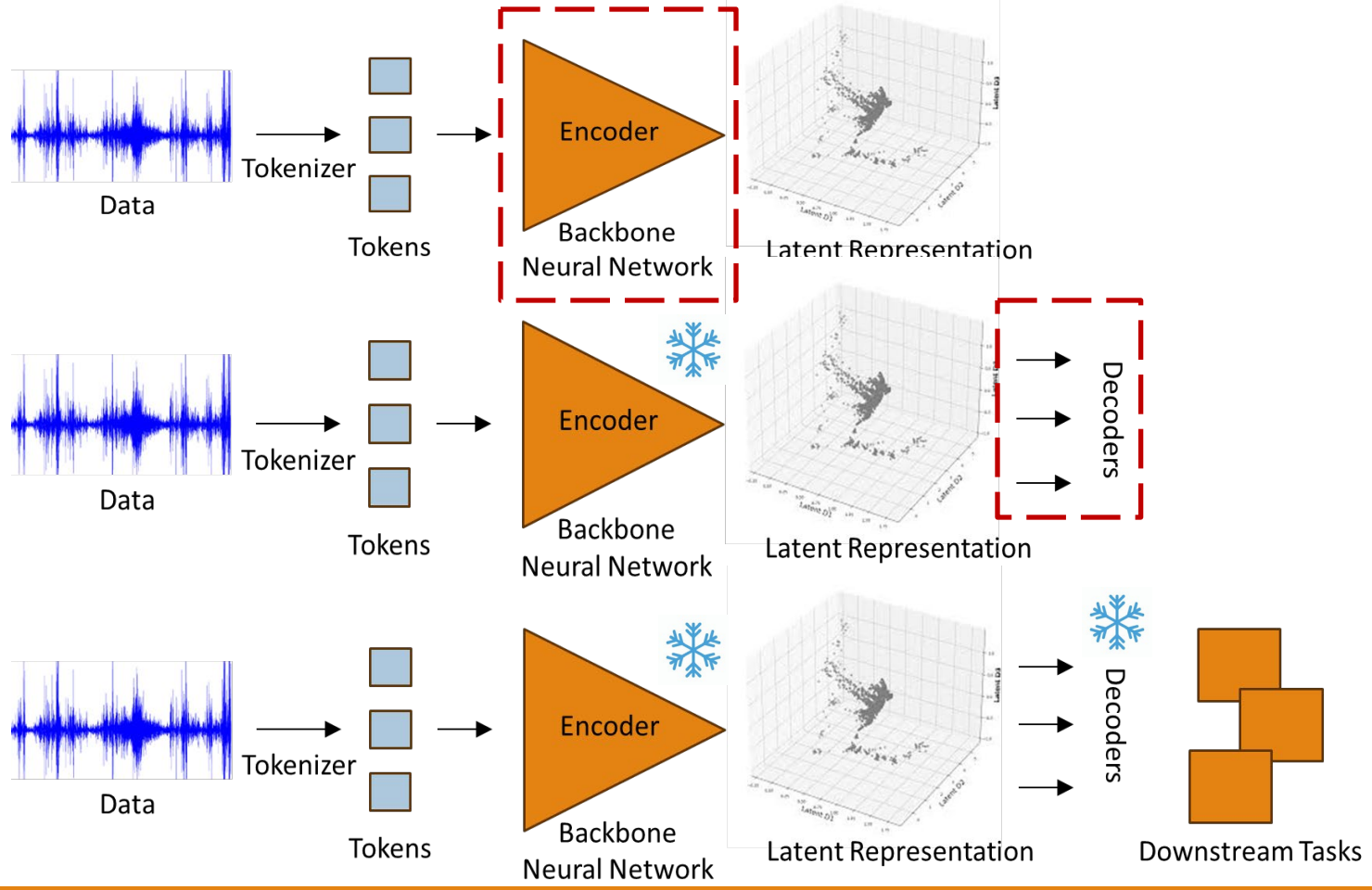
- The neural network that converts the stream of input tokens into a latent representation

Part III: The pretext task(s)

- The task that trains the encoder



Stages of Self-Supervised Learning



Training (with pretext task)

Fine-Tuning (supervised)

Testing/Deployment

Distinctive Properties of Self-Supervised Learning

- How does it differ from supervised learning?

Distinctive Properties of Self-Supervised Learning

- How does it differ from supervised learning?
  - **Supervised:** Uses labels to teach specific tasks
  - **Self-supervised:** Learns “data representation” without labels.

Distinctive Properties of Self-Supervised Learning

- How does it differ from supervised learning?
  - **Supervised:** Uses labels to teach specific tasks
  - **Self-supervised:** Learns “data representation” without labels.
- How does it different unsupervised learning (e.g., clustering, principal component analysis, etc)?

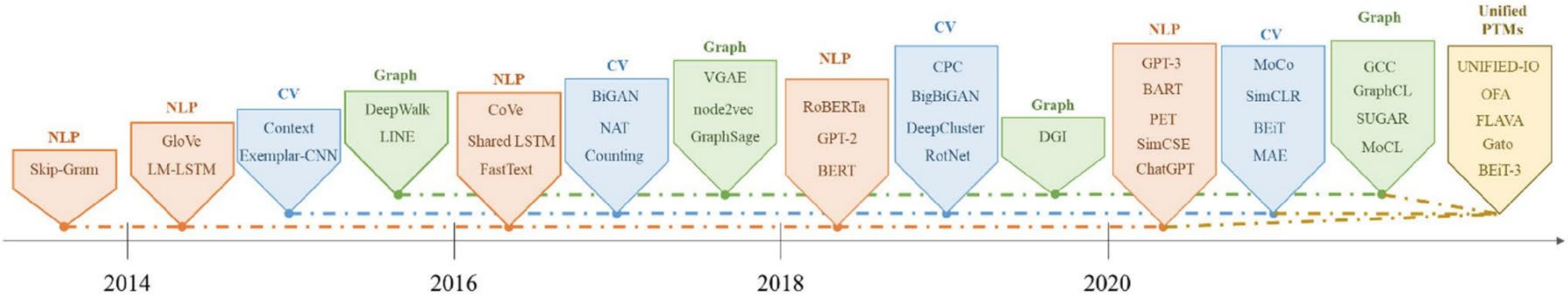
Distinctive Properties of Self-Supervised Learning

- How does it differ from supervised learning?
  - **Supervised:** Uses labels to teach specific tasks
  - **Self-supervised:** Learns “data representation” without labels.
- How does it different unsupervised learning (e.g., clustering, principal component analysis, etc)?
  - **Pretext Task Definition:** Self-supervised learning defines a pretext task (e.g., predicting the next word, rotating an image) to learn, while unsupervised learning focuses on algorithm for finding structure (e.g., clustering).
  - **Goal:** Self-supervised learning aims to learn a high-quality “feature space” (or latent representation of input data) to be used for downstream tasks. Unsupervised learning usually relies on a pre-existing feature space definition to be used directly for structuring the data (e.g., predefined distance metrics for clustering).

History of Self-Supervised Learning and Foundation Models

[Historical Fact] Three key driving domains:

- Natural Language Processing (NLP)
- Computer Vision (CV)
- Graph Analytics (Graph)



https://link.springer.com/article/10.1007/s13042-024-02443-6

Components of Self-Supervised Learning

Part I: The tokenizer

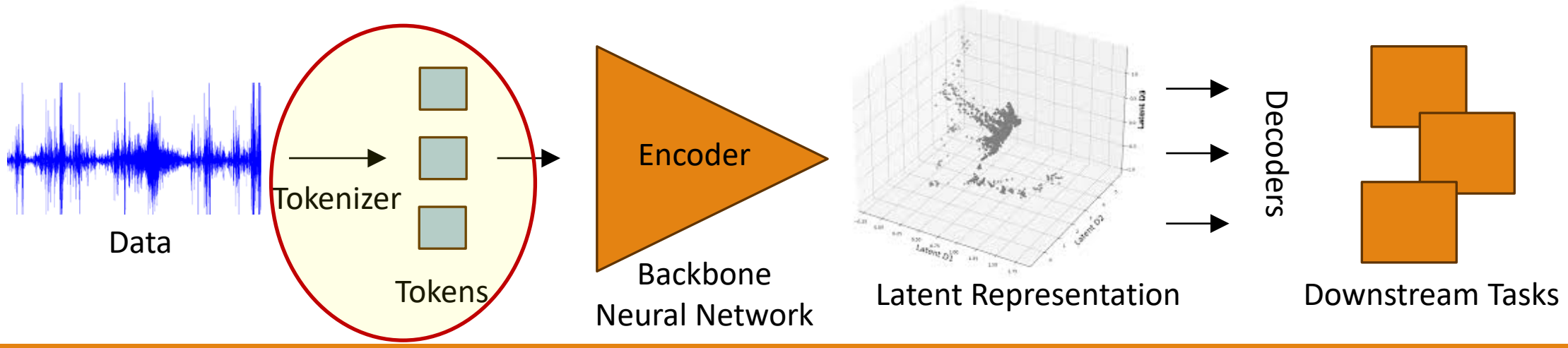
- Breaks the input stream into pieces to be individually encoded.

Part II: The backbone neural network model (or encoder)

- The neural network that converts the stream of input tokens into a latent representation

Part III: The pretext task(s)

- The task that trains the encoder



SSL Part I: The Tokenizer (NLP)

1. Break text by word.

- Issues?

SSL Part I: The Tokenizer (NLP)

1. Break text by word.

- Pros:
  - Easy to implement
- Cons:
  - Large vocabulary
  - Similar words may have different representations (e.g., “it is” and “it’s”)
  - Some languages do not use spaces

SSL Part I: The Tokenizer (NLP)

1. Break text by word.

- Pros:
  - Easy to implement
- Cons:
  - Large vocabulary
  - Similar words may have different representations (e.g., “it is” and “it’s”)
  - Some languages do not use spaces

- **How to mitigate the cons?**

SSL Part I: The Tokenizer (NLP)

2. Subword tokenization: Split words into common subwords

- Pros:
  - Fewer building blocks
  - Rare words could be decomposed into more common subwords
- Examples:
  - WordPiece: Used in BERT
    - Uses a greedy, longest-match-first strategy to segment text based on a pre-trained vocabulary
  - BPE: Used in ChatGPT
    - Starting with individual characters, uses frequent pattern mining to find the most frequent words/subwords

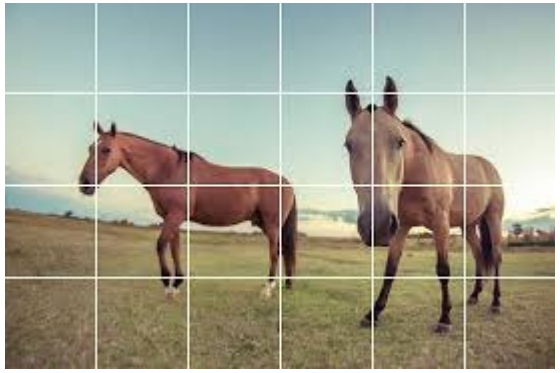
SSL Part I: The Tokenizer (NLP)

2. Subword tokenization: Split words into common subwords

- Pros:
  - Fewer building blocks
  - Rare words could be decomposed into more common subwords
- Examples:
  - WordPiece: Used in BERT
    - Uses a greedy, longest-match-first strategy to segment text based on a pre-trained vocabulary
  - BPE: Used in ChatGPT
    - Starting with individual characters, uses frequent pattern mining to find the most frequent words/subwords
- Positional Embedding:
  - After tokenizing, each token has a position in the input stream. Positional embedding encodes the token's position allowing it to contribute to overall semantics:
    - For example: "Dog bit girl" versus "girl bit dog"

SSL Part I: The Tokenizer (CV)

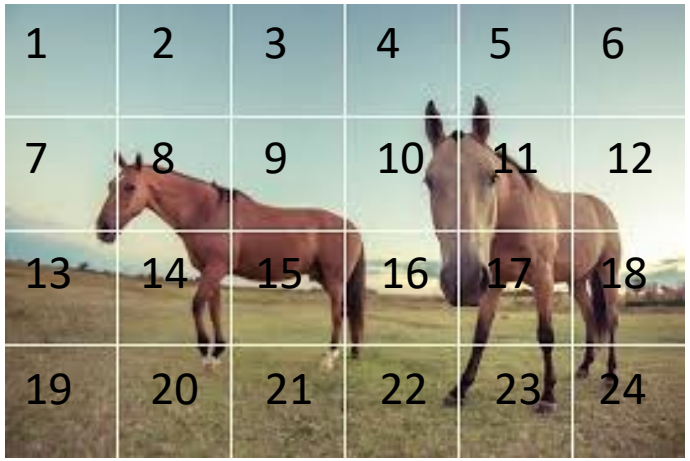
Break image into patches.



Example: ViT
https://github.com/lucidrains/vit-pytorch

SSL Part I: The Tokenizer (CV)

Break image into patches.



Patches have a positional embedding that encodes their position inside the picture

Example: ViT
https://github.com/lucidrains/vit-pytorch

SSL Part I: The Tokenizer (Graph)

Node-Level Tokenization: Individual nodes are represented by tokens. Token embedding encode relation to neighboring nodes (a kind of positional embedding)

Graph-Level/Subgraph Tokenization: The graph is broken down into motifs or subgraphs, which are then mapped to fixed-length tokens.

Differentiable Graph Tokenization:A Neural Network acts as a tokenizer to convert node representations into a latent token vocabulary.

…

SSL Part I: The Tokenizer (IoT) – Open Research Challenge

Fixed-length Tokenization: Each batch of n samples is converted into a token

Issues?

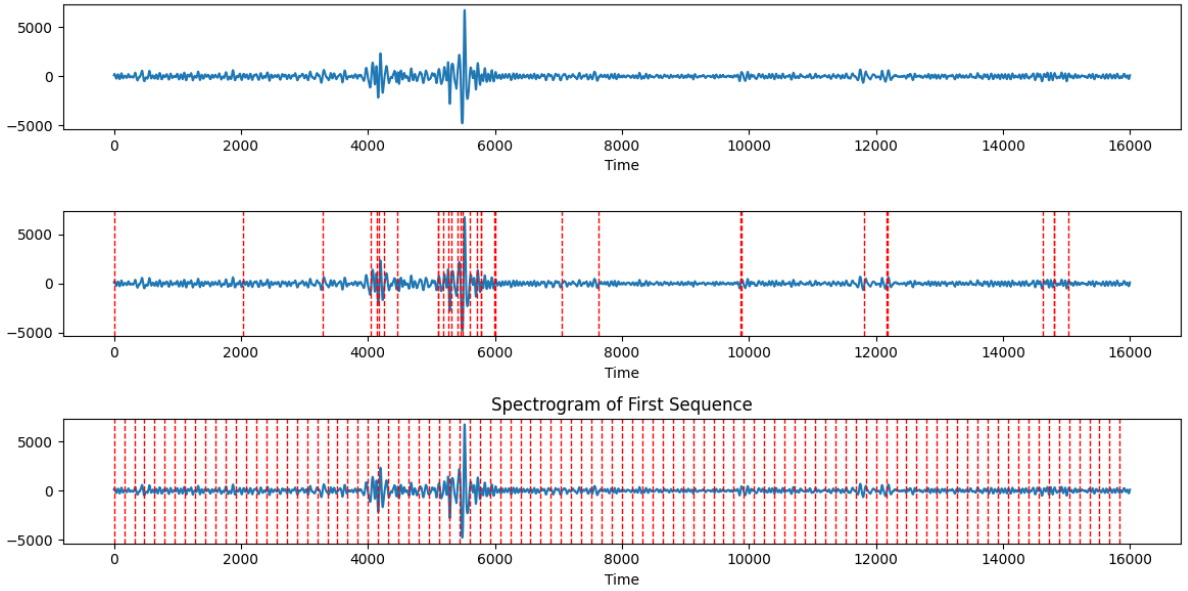
SSL Part I: The Tokenizer (IoT) – Open Research Challenge

Fixed-length Tokenization: Each batch of n samples is converted into a token

Variable-length Tokenization: Samples are clustered into groups of different sizes depending on latent information content. Each cluster is converted into a token.

Variable length Tokenization

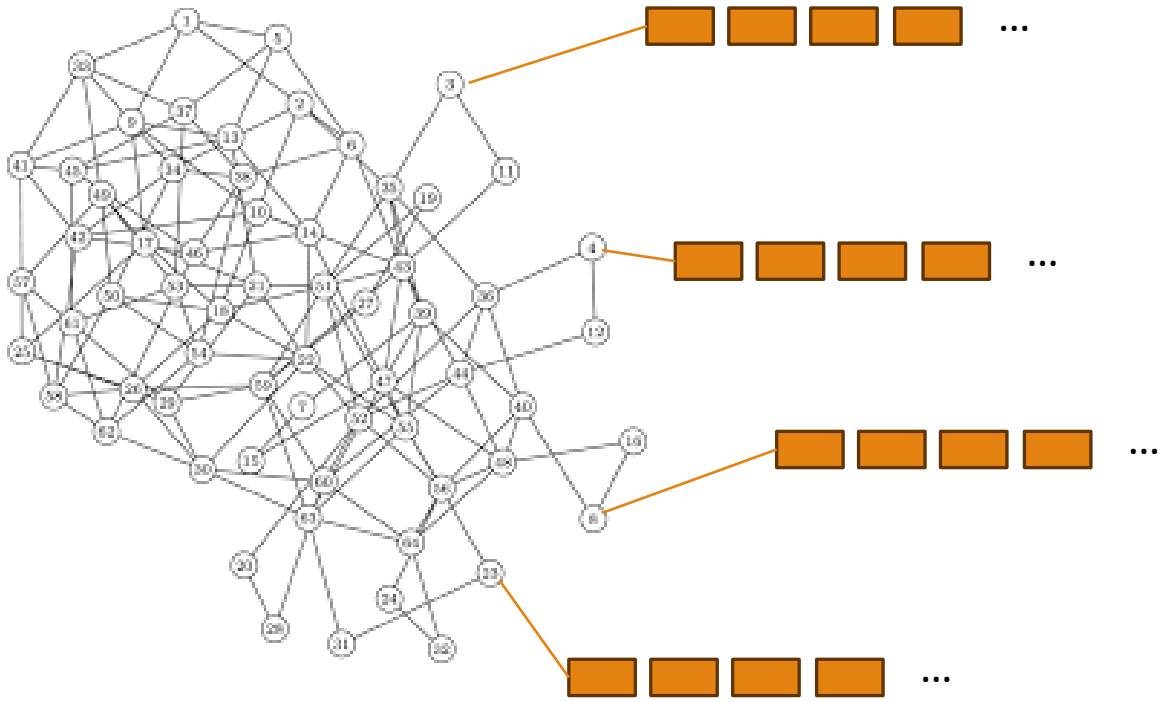
Fixed length Tokenization



Picture generated by Tommy Kimura

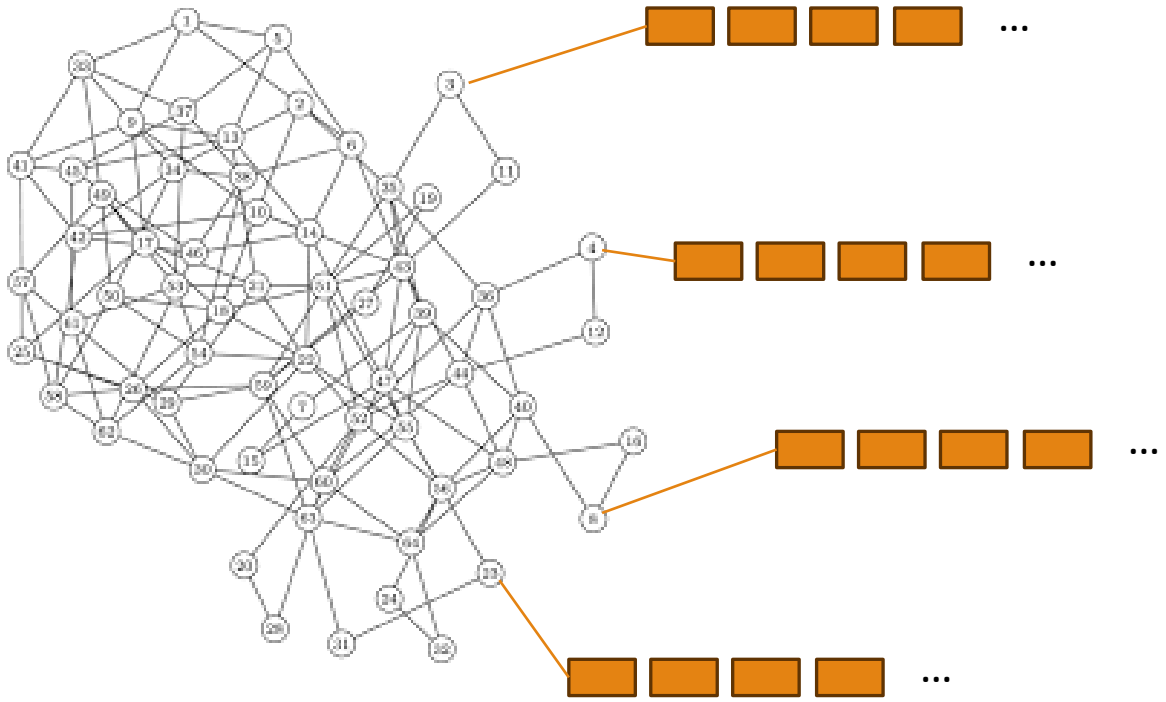
SSL Part I: The Tokenizer (IoT) – Open Research Challenge

Tokenization of distributed streams?



SSL Part I: The Tokenizer (IoT) – Open Research Challenge

Tokenization of distributed streams?



How to learn positional embedding of token stream sources (vantage point embedding)?

Inspirations from graphs?

Components of Self-Supervised Learning

Part I: The tokenizer

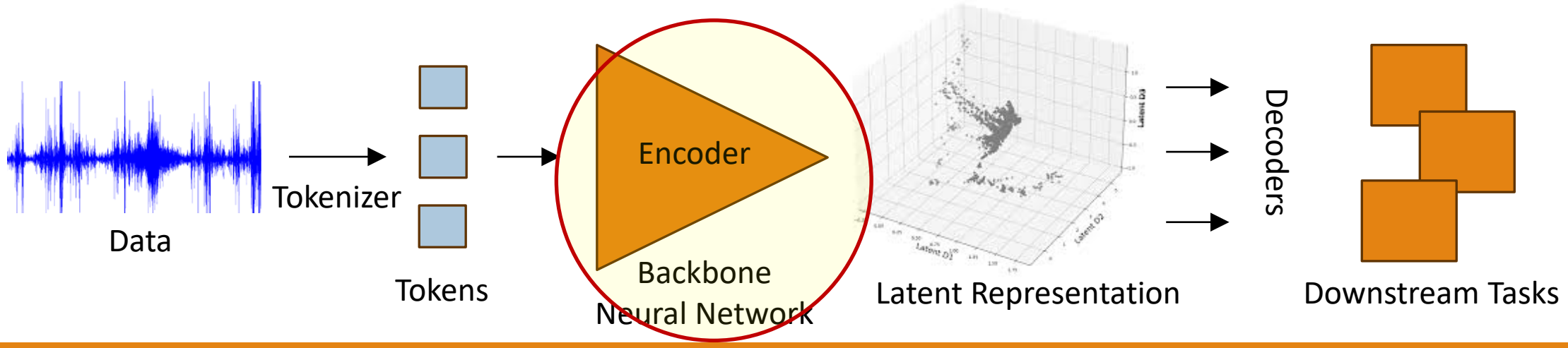
- Breaks the input stream into pieces to be individually encoded.

Part II: The backbone neural network model (or encoder)

- The neural network that converts the stream of input tokens into a latent representation

Part III: The pretext task(s)

- The task that trains the encoder

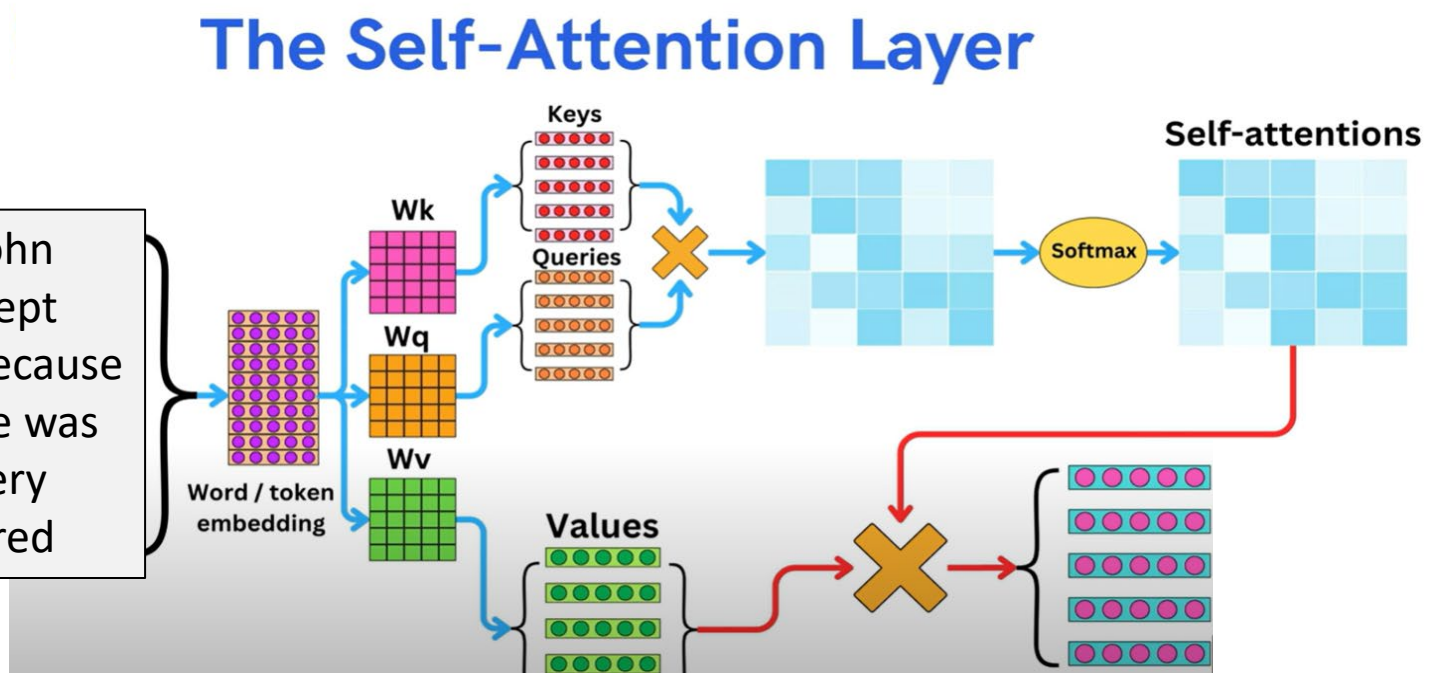


SSL Part II: The Backbone Neural Network Model (Encoder)

- **In principle, it can be any neural network that transforms input into an embedding (latent representation) can function as an encoder.**
- **Examples include:**
  - *Convolutional Neural Networks (CNNs):* An example is ResNet (extracts features from image patches).
  - *Transformers:* An example is Vision Transformer (ViT), used in Masked Autoencoders (MAE), to encode image patches.
  - *Siamese Encoders:* Used in contrastive learning to process different views of the same data, such as in SimCLR.
  - *Autoencoders:* Variational Auto-Encoders (VAEs) or convolutional autoencoders that compress data to learn latent representations.

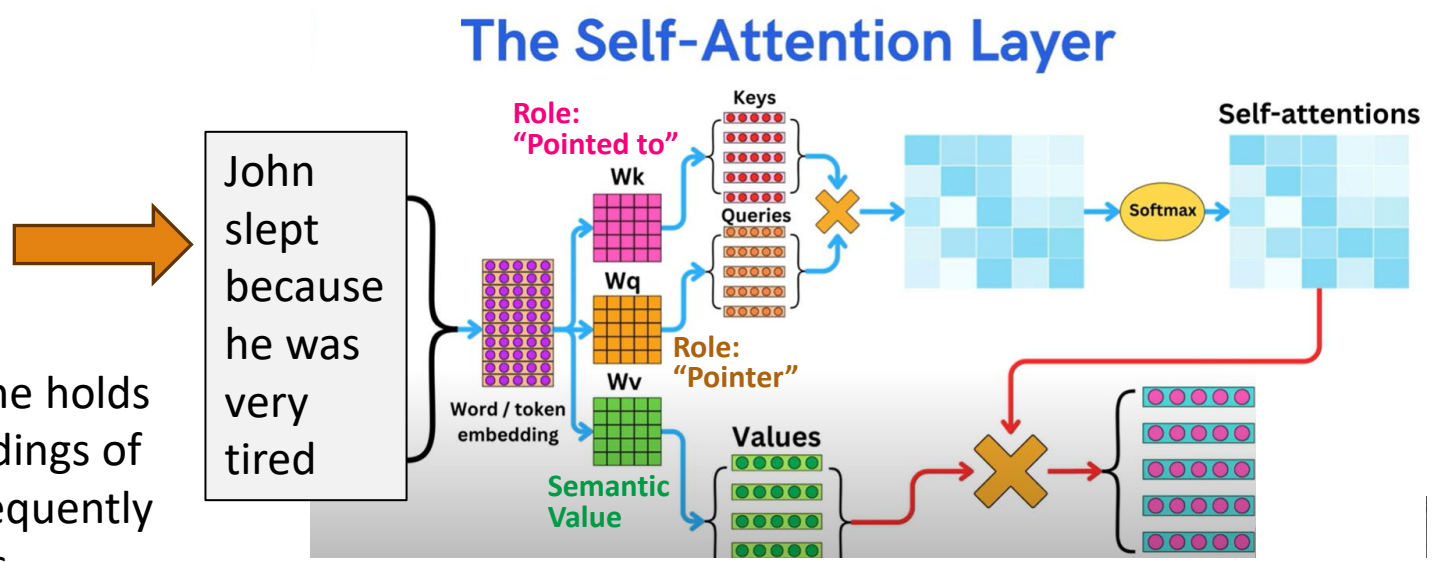
SSL Part II: The Backbone Neural Network Model (Encoder)

- **Example: Transformers** (Attention is all you need):
  - https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

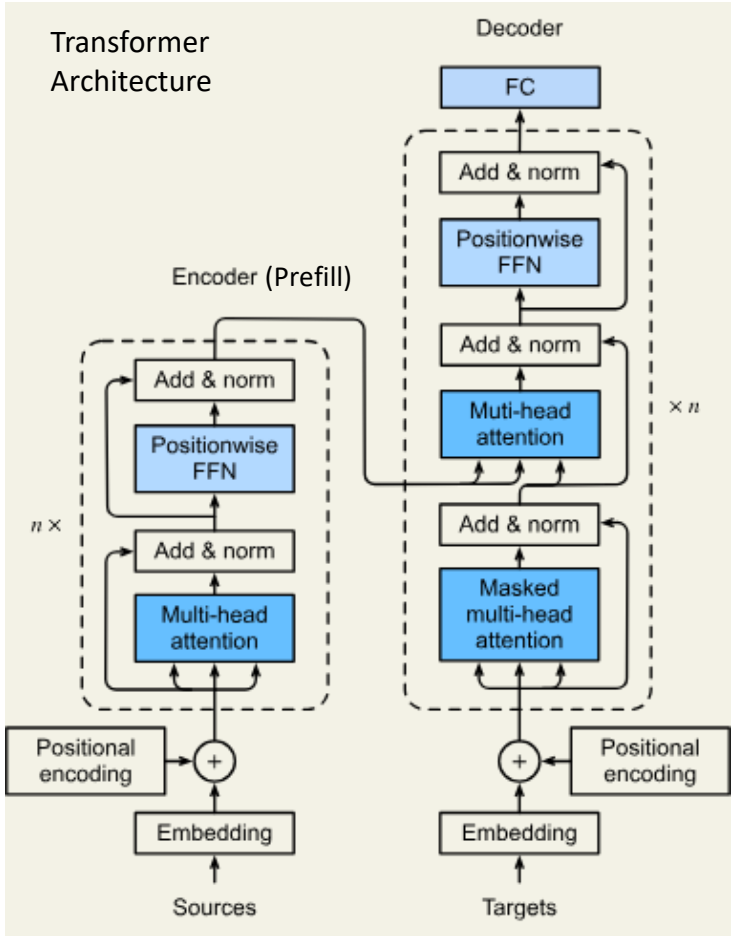


SSL Part II: The Backbone Neural Network Model (Encoder)

- **Transformers** (Attention is all you need)

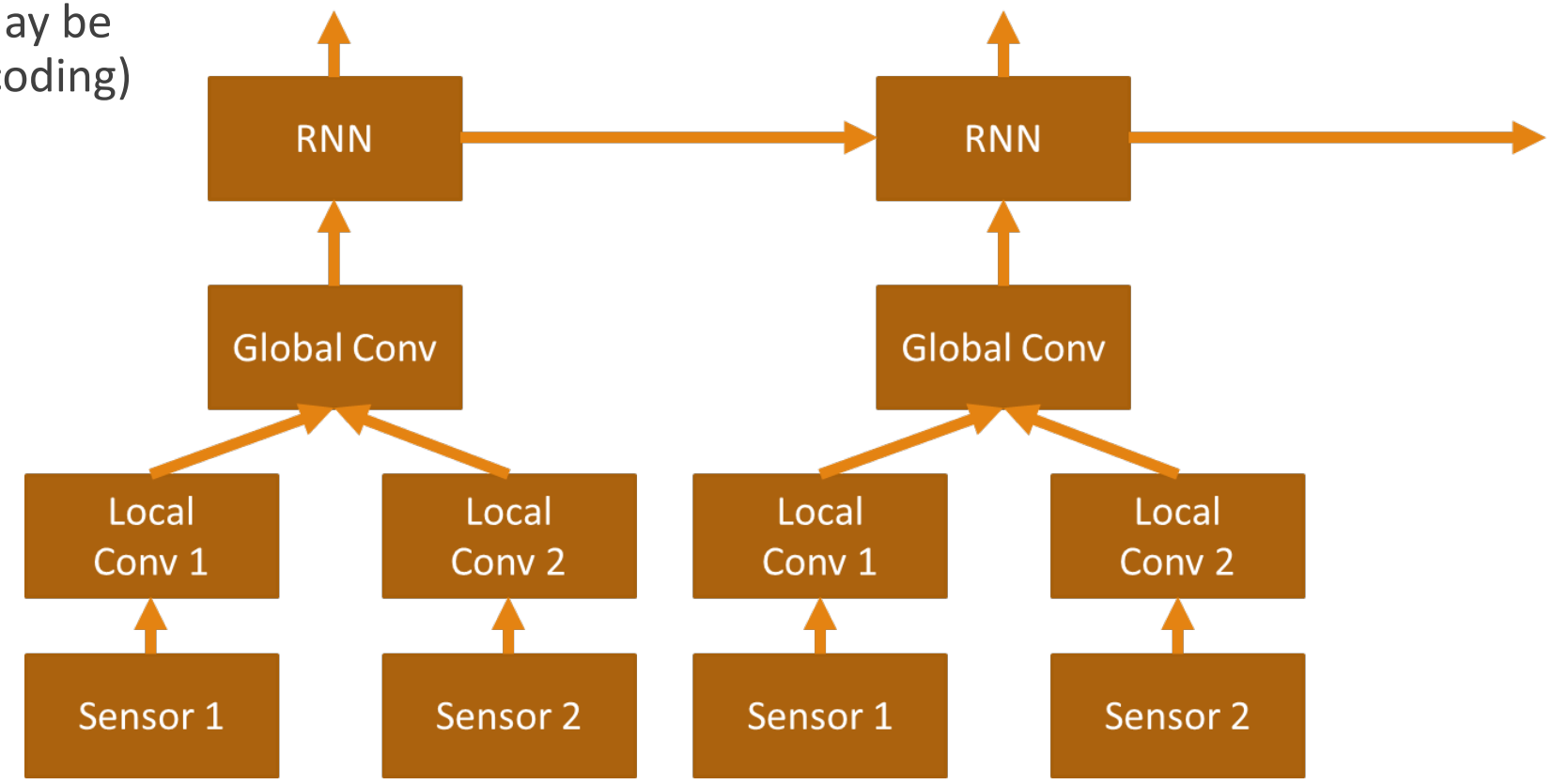


The KV cache holds the embeddings of recently/frequently used tokens



SSL Part II: The Backbone Neural Network Model (Encoder)

- **Example: CNNs/RNNs** (May be sufficient for IoT data encoding)
  - Used for IoT data



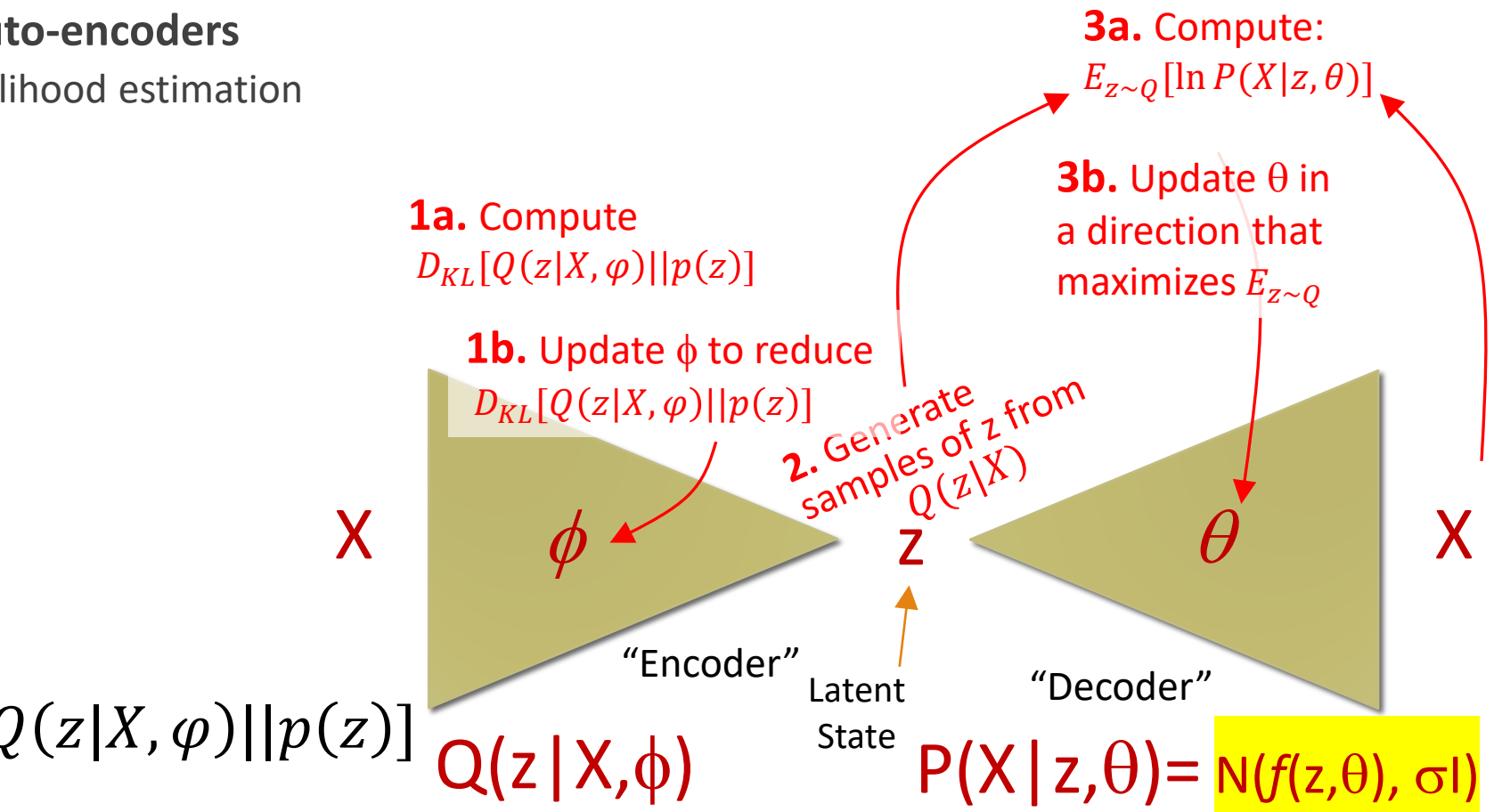
DeepSense

SSL Part II: The Backbone Neural Network Model (Encoder)

- **Example: Variational Auto-encoders**
  - Based on maximum likelihood estimation

Good objective function!!

$$Obj(\theta, \varphi) = E_{z\sim Q}[\ln P(X|z,\theta)] - D_{KL}[Q(z|X,\varphi)||p(z)]$$



A Simple Estimation Problem

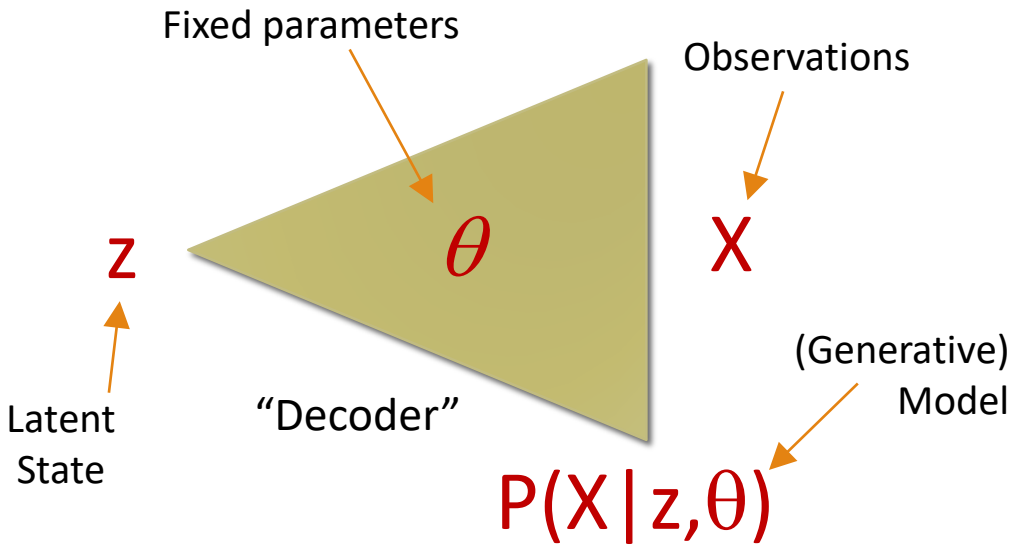
The system:

A system produces observations, X, from a generative process that depends on some latent state, z, and some parameters, θ.

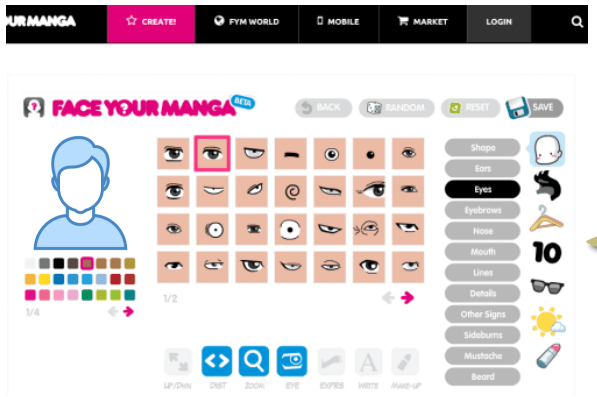
The generative model:

$p(X|z,\theta)$ = ✓

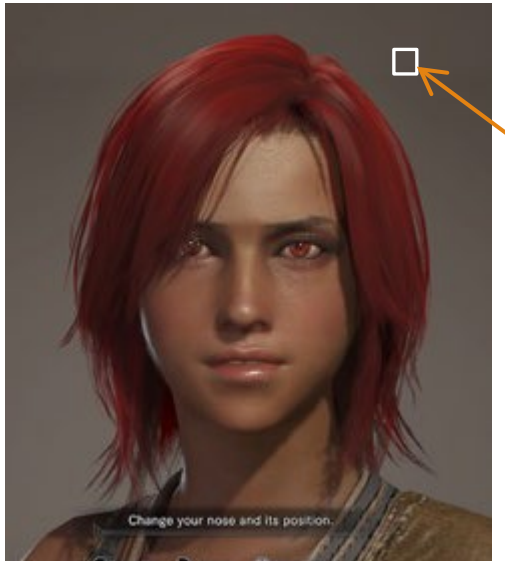
The challenge: Find z, θ.



Example of a Generative Process



Transformation (decoding), θ.



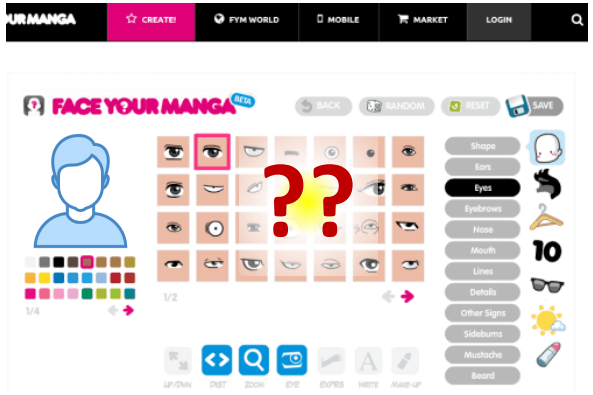
Note: The color of each "decoded" pixel can be determined from the latent variables independently from other pixels. (Conditional independence.)

Data sample (face), **X**.

$P(X|z,\theta)$

The Estimation Problem is the Inverse of the Generative Process

Encoder



Given a lot of examples, **x**

find: z, θ

Problem: System produces observations, X, from a process that depends on latent state, z, and fixed parameters, θ. Given X, jointly estimate z, and θ.

Expectation Maximization

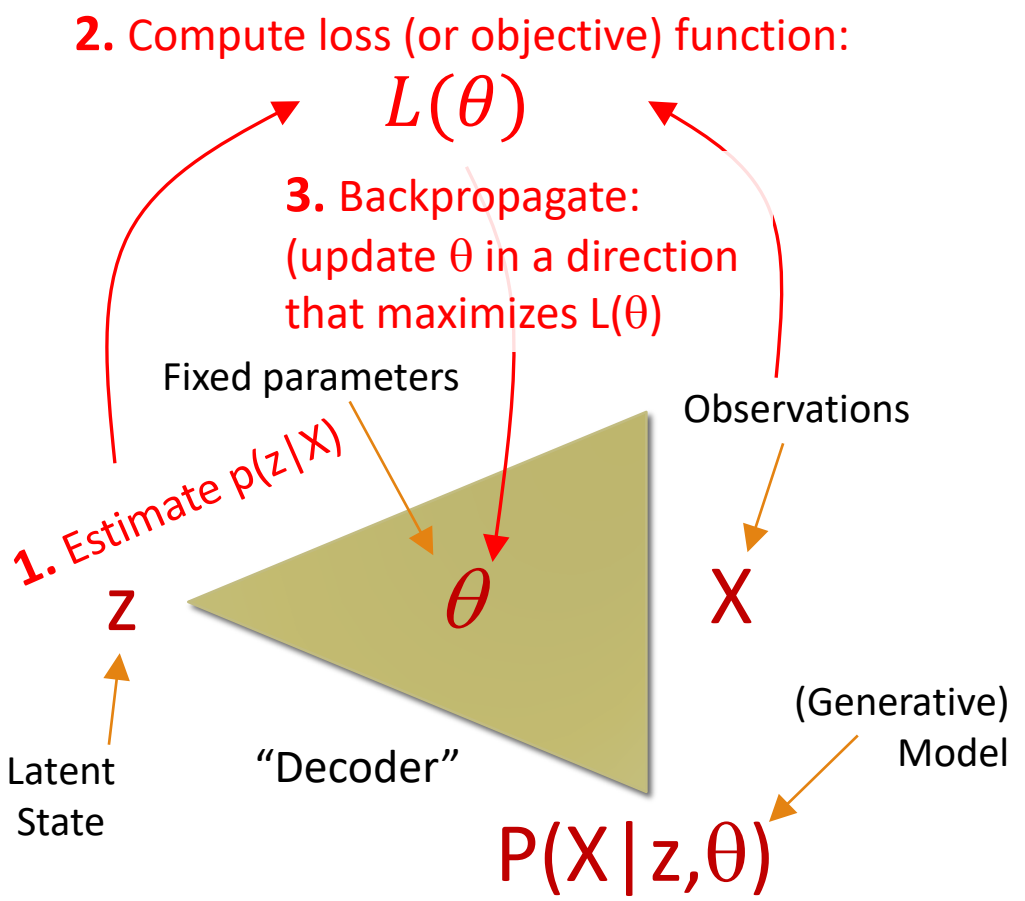
Step 1: Estimate the "posterior" (distribution of latent variables given the observations):

$$p(z|X,\theta) = \frac{p(X|z,\theta)p(z|\theta)}{p(X|\theta)} = \frac{p(X|z,\theta)p(z)}{\int_z p(X|z,\theta)p(z)}$$

Step 2: Compute "loss function" or objective function (in EM, the expected log likelihood):

$$L(\theta) = E_{z\sim p(z|X,\theta)} \ln p(X,z|\theta)$$

Step 3: "Back-propagate" or maximize with respect to θ. $dL(\theta)/d\theta = 0 \rightarrow \theta =$ ✓



Problem: System produces observations, X, from a process that depends on latent state, z, and fixed parameters, θ. Given X, jointly estimate z, and θ.

Expectation Maximization

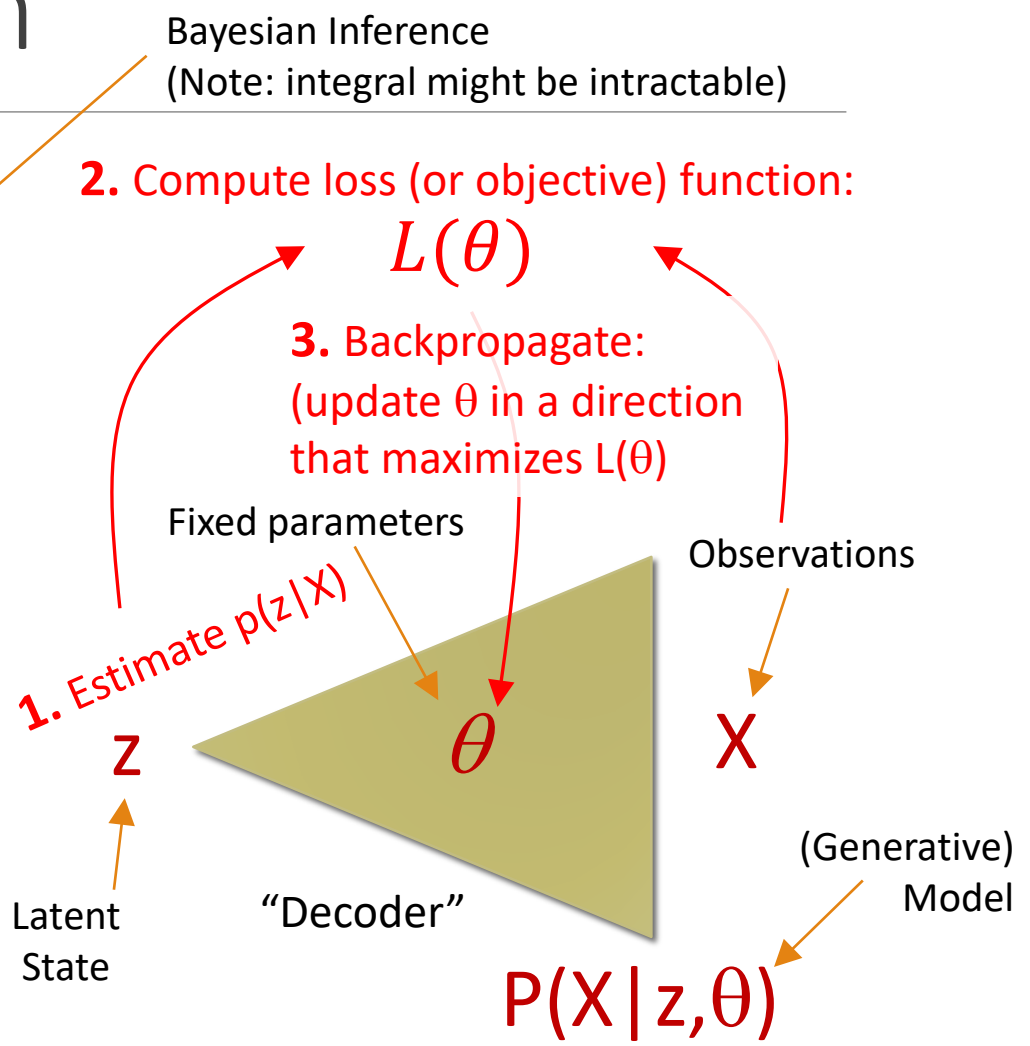
Step 1: Estimate the "posterior" (distribution of latent variables given the observations):

$$p(z|X,\theta) = \frac{p(X|z,\theta)p(z|\theta)}{p(X|\theta)} = \frac{p(X|z,\theta)p(z)}{\int_z p(X|z,\theta)p(z)}$$

Step 2: Compute "loss function" or objective function (in EM, the expected log likelihood):

$$L(\theta) = E_{z\sim p(z|X,\theta)} \ln p(X,z|\theta)$$

Step 3: "Back-propagate" or maximize with respect to θ. $dL(\theta)/d\theta = 0 \rightarrow \theta =$ ✓



Problem: System produces observations, X, from a process that depends on latent state, z, and fixed parameters, θ. Given X, jointly estimate z, and θ.

Variational Inference

Challenge: What if $p(z|X,\theta)$ is intractable?

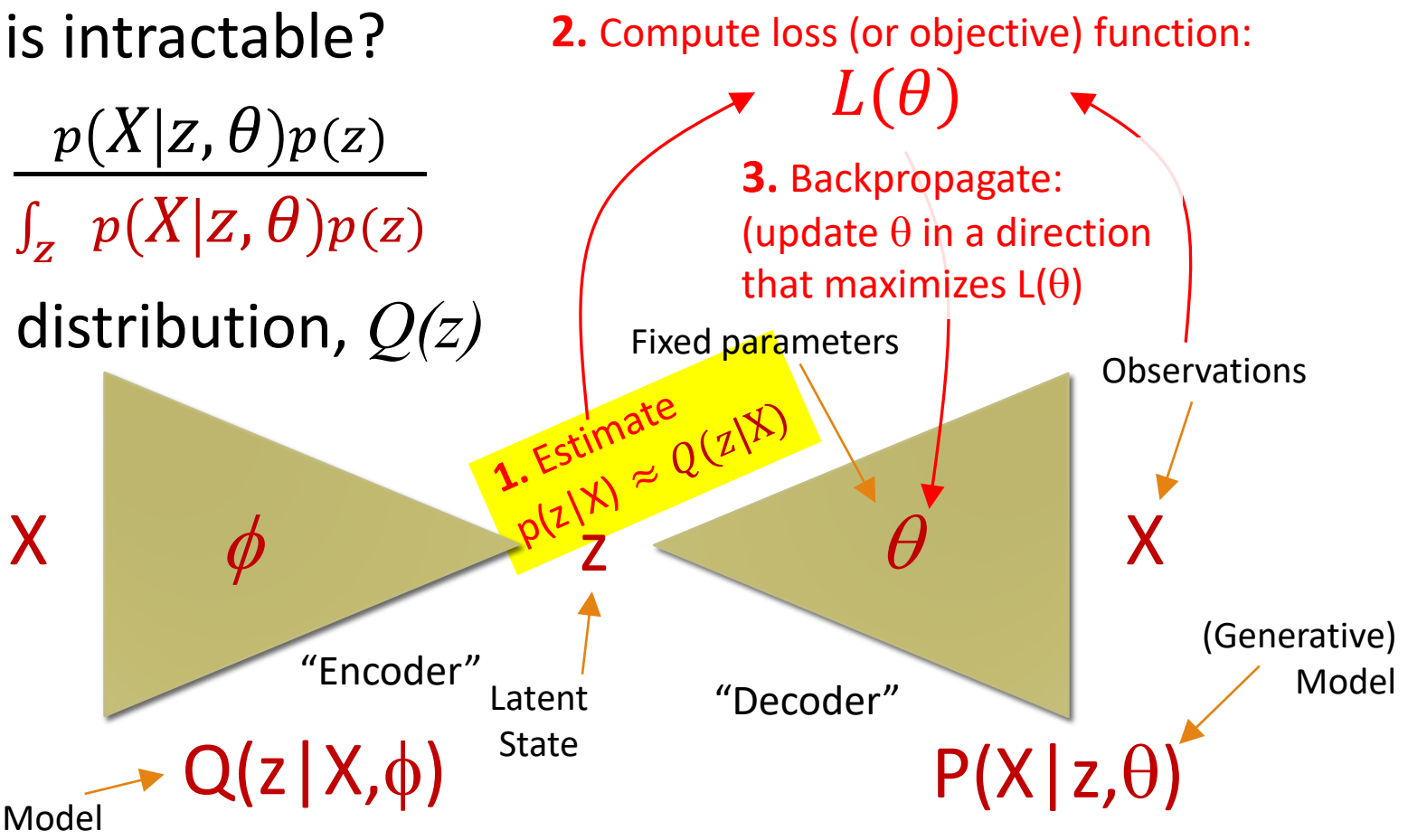
$$p(z|X,\theta) = \frac{p(X|z,\theta)p(z|\theta)}{p(X|\theta)} = \frac{p(X|z,\theta)p(z)}{\int_z p(X|z,\theta)p(z)}$$

Solution: Approximate it by distribution, $Q(z)$

$$Q(z|X,\varphi) \approx p(z|X,\theta)$$

That minimizes:

$$D_{KL}\left(Q(z|X,\varphi)||p(z|X,\theta)\right)$$



Problem: System produces observations, X, from a process that depends on latent state, z, and fixed parameters, θ. Given X, jointly estimate z, and θ.

The Variational Auto-Encoder (VAE)

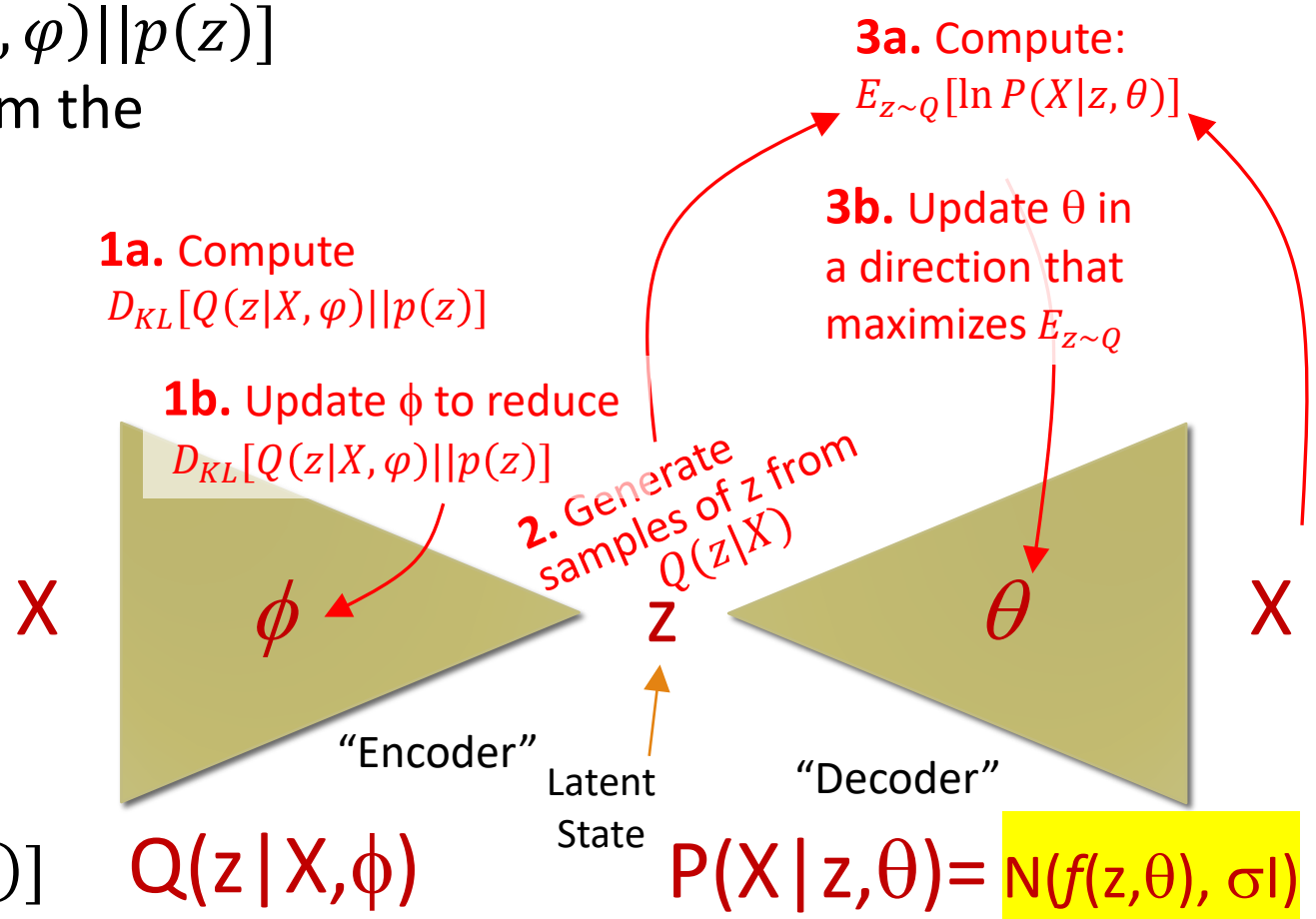
Step1: SGD: Adapt φ to reduce $D_{KL}[Q(z|X,\varphi)||p(z)]$

Step2: Generate multiples samples of z from the distribution N(μ(X,ϕ), σ(X,ϕ))

Step3: SGD: Adapt θ to increase $P(X|z,\theta)$

Good objective function!!

$$Obj(\theta,\varphi) = E_{z\sim Q}[\ln P(X|z,\theta)] - D_{KL}[Q(z|X,\varphi)||p(z)]$$



Components of Self-Supervised Learning

Part I: The tokenizer

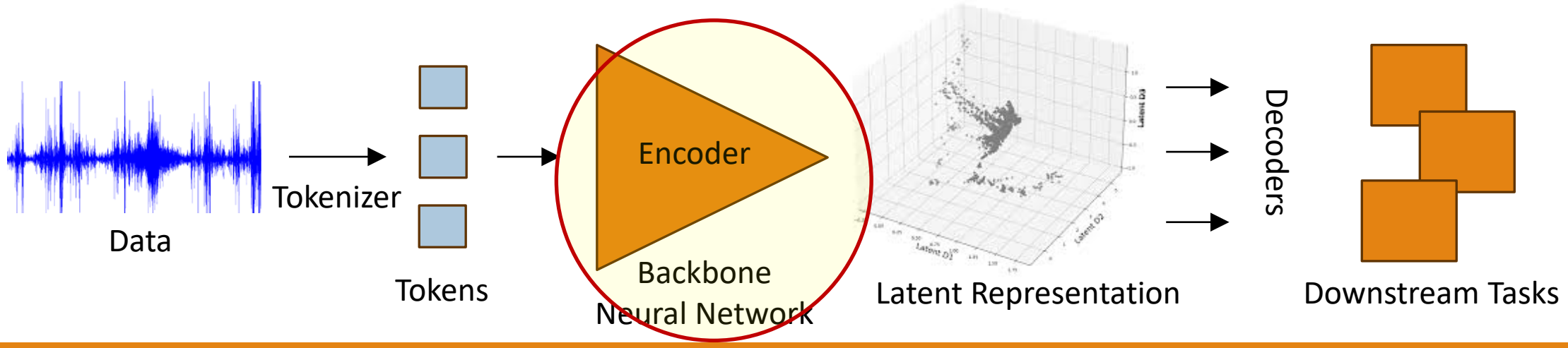
- Breaks the input stream into pieces to be individually encoded.

Part II: The backbone neural network model (or encoder)

- The neural network that converts the stream of input tokens into a latent representation

Part III: The pretext task(s)

- The task that trains the encoder



SSL Part III: Pretext Tasks

Artificial tasks that create training signals:

- Reconstruction (similar to VAE)
- Masking
- Prediction (of next token, future trajectory, etc)
- Contrastive learning

NLP Pretext Tasks

- **Thoughts?**

NLP Pretext Tasks

- **Mask language modeling (MLM)**
  - Erases some words randomly in the input sequence and then predicts these erased words during pretraining (e.g., BERT)
- **Denoising autoencoder (DAE)**
  - Adds noise to the original corpus and reconstruct the original input from the corpus containing noise (e.g., BART)
- **Replaced token detection (RTD)**
  - Determines whether the current token is original or replaced (e.g., ELECTRA)

Computer Vision Pretext Tasks

- **Thoughts?**

Computer Vision Pretext Tasks

- **Masked data reconstruction**
  - The original image is first divided into a few patches and discrete visual tokens are used to encode each patch. Some patches are masked and their tokens predicted.
- **Frame order learning**
  - Learns frame order from videos

Graph Pretext Tasks

- **Thoughts?**

Graph Pretext Tasks

- **Graph information completion**
  - Masking part of the information in the input graph, and then recovering the masked information based on the analysis of the remaining information distribution.
- **Graph property prediction**
  - For example, assigning pseudo-labels to nodes through clustering, and predicting node clusters.
- **Graph reconstruction**
  - Ensuring that the latent representation is sufficient to reconstruct the original graph
- **Contrastive learning**
  - Ensure proximity (in the latent space) between similar nodes and distance between dissimilar ones.

Covered Today: Components of Self-Supervised Learning

Part I: The tokenizer

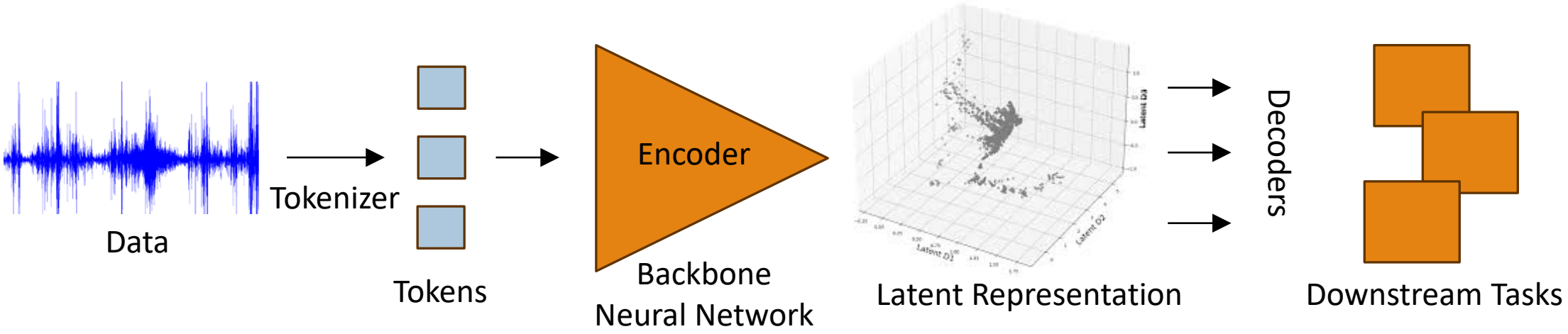
- Breaks the input stream into pieces to be individually encoded.

Part II: The backbone neural network model (or encoder)

- The neural network that converts the stream of input tokens into a latent representation

Part III: The pretext task(s)

- The task that trains the encoder



Moral of the Story

- Adaptation of self-supervised learning to a new application domain starts with asking three key questions:
  - How do I tokenize data from that domain into the most appropriate building blocks, well-suited for efficient semantic representation?
  - What kind of encoder architecture do I need to encode such tokens into an appropriate latent space for the domain?
  - What pretext (i.e., training) tasks do I use for training the encoder to best represent domain data to enable a broad range of downstream analytics?
- Examples of answers to the above three questions were given for the domains of natural language processing, computer vision, graph analytics, and IoT.
- In general, different IoT applications have their own data characteristics that call for novel application-specific self-supervised learning frameworks… this is an area of active research.