Attention and versatile architectures

Presenter: Seunghoon Hong

Course overview

- Image classification
- Object detection
- Semantic segmentation
- Visualization
- Style transfer
- Adversarial attacks

- Text modeling
- Machine translation
- Image captioning
- Visual question answering

- Image generation
- Text generation
- Img-to-img translation

- Attention and versatile networks
- Self- and Semi-supervised learning
- Multi-modal learning
- Graph neural networks

We are here!

Convolutional Neural Networks (CNN)

Recurrent Neural Networks (RNN)

Deep generative models

Advanced topics

So far, we learned various neural networks

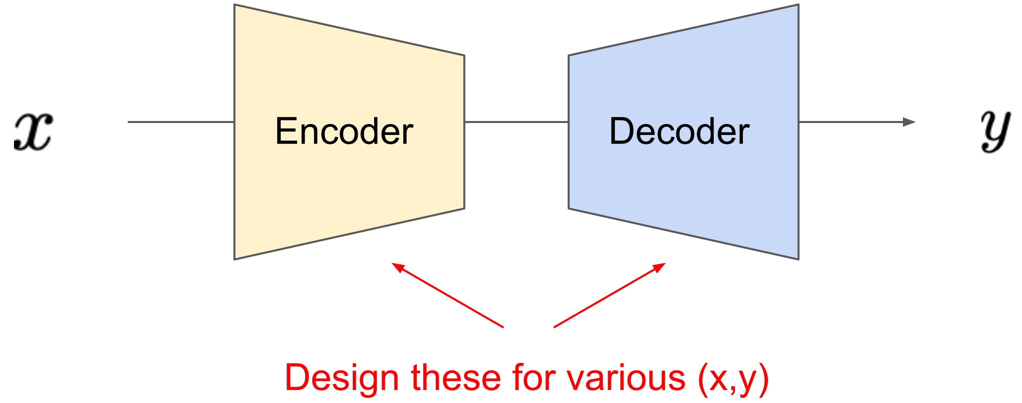
- Multi-Layer Perceptron (MLP)
  - Linear layer + Non-linear function
- Convolutional Neural Network (CNN)
  - Convolution + Pooling + Non-linear function + MLP
- Recurrent Neural Network (RNN)
  - MLPs for recurrent update + gating functions
- Attention and Transformer
  - Dot-product attention + MLPs

We also learned about various tasks

- Classification
- Detection
- Segmentation
- Text encoding
- Machine translation
- Image captioning
- Visual question answering
- …

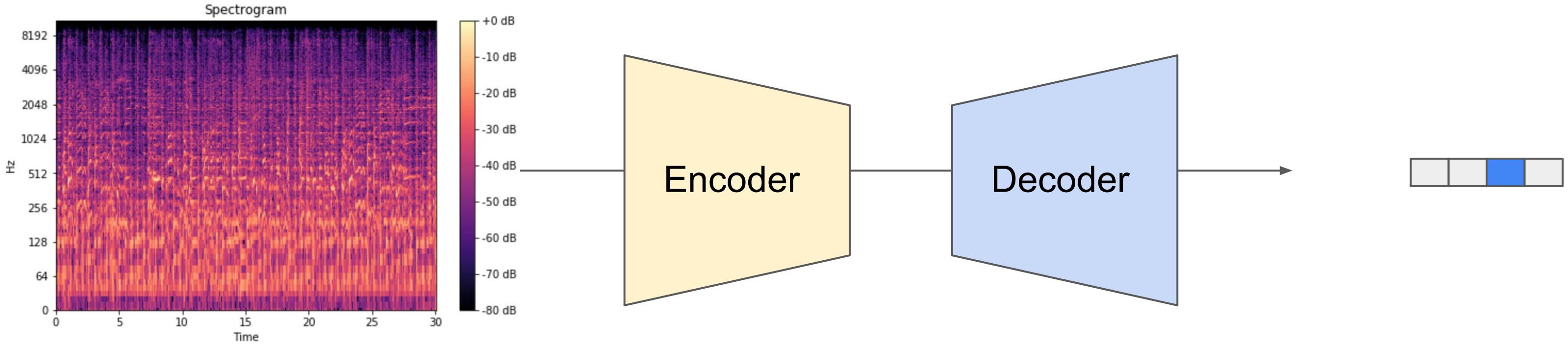
We are ready to be DL practitioners

- Let's try solving some problems!



Practice 1: speaker identification

Given a speech signal, identify the speaker given the list of known people.

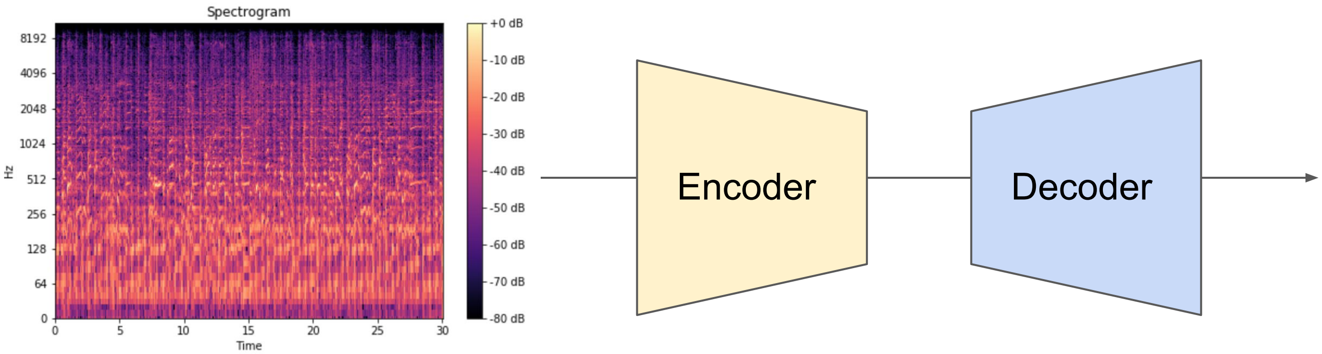


X: Mel-spectogram

Y: {Yumi, Jack, **John**, Jane}

Practice 2: speech to text translation

Given a speech signal, transcribe the content into the text



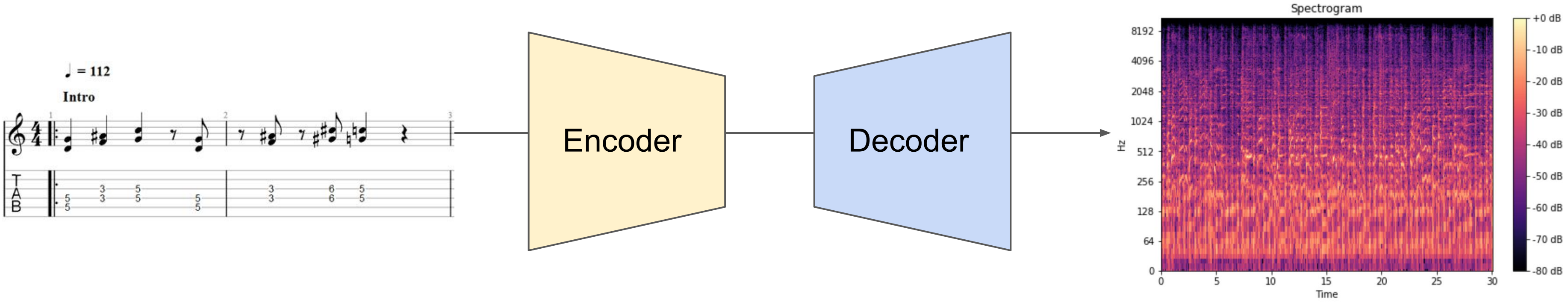
"Siri, turn off the light"

X: Mel-spectogram

Y: natural language

Practice 3: creating music from score

Given the music score, create the piano song playing the score

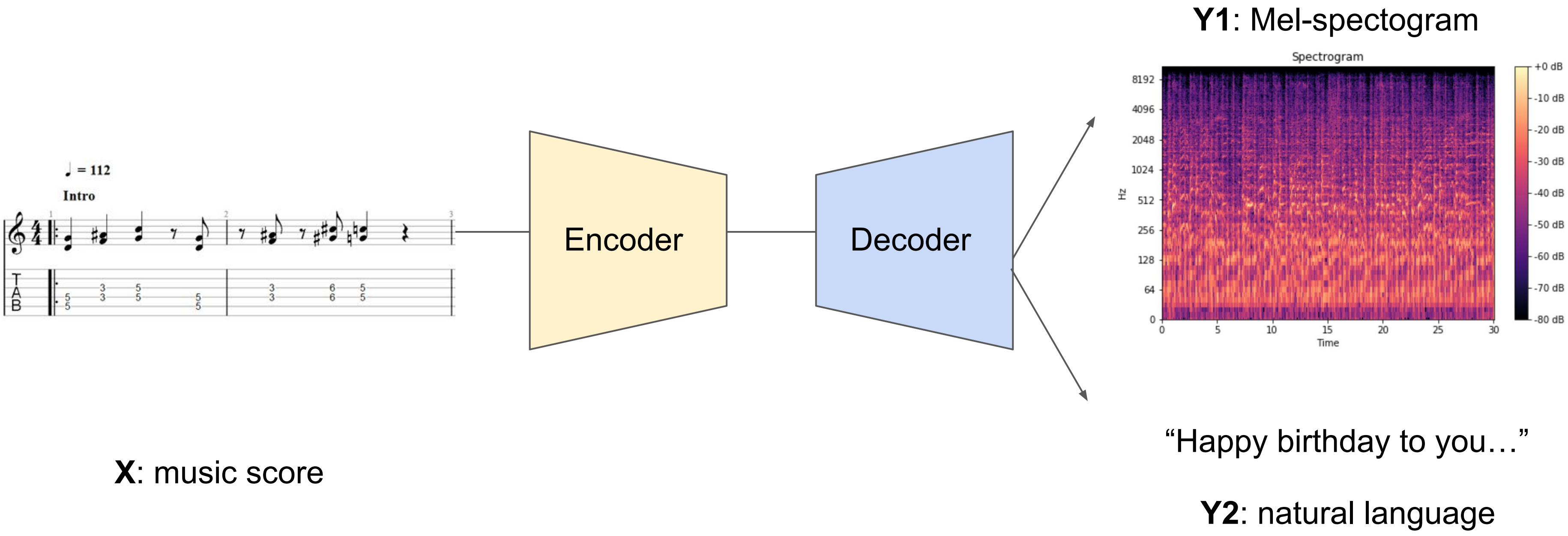


X: music score

Y: Mel-spectogram

Problem 4: creating music and lyrics from score

Given the music score, create the piano song playing the score and lyrics



Different data (tasks), different models

- Why do we have different models for different data and tasks?

- What happens if we apply different combinations of data and models?
    - Example 1: CNN for sequences.
    - Example 2: RNN for images.
    - Example 3: MLP for images.

Inductive bias in neural networks

- **Inductive bias (formal definition)**: a set of assumptions that the learner uses to predict outputs given input that has not observed

Inductive bias in neural networks

- **Inductive bias (formal definition)**: a set of assumptions that the learner uses to predict outputs given input that has not observed

- Many neural network architectures rely on certain type of inductive biases
  - Mostly depends on locality and invariance of data
  - Example1: Convolution exploits 2D grid data structure and translation invariance
  - Example 2: RNN exploits 1D chain data structure and first-order Markov assumption

Inductive bias in neural networks

- **Inductive bias (formal definition)**: a set of assumptions that the learner uses to predict outputs given input that has not observed

- Many neural network architectures rely on certain type of inductive biases
  - Mostly depends on locality and invariance of data
  - Example1: Convolution exploits 2D grid data structure and translation invariance
  - Example 2: RNN exploits 1D chain data structure and first-order Markov assumption

- These assumptions are generally useful to design efficient models

Inductive bias in neural networks

- **Inductive bias (formal definition)**: a set of assumptions that the learner uses to predict outputs given input that has not observed

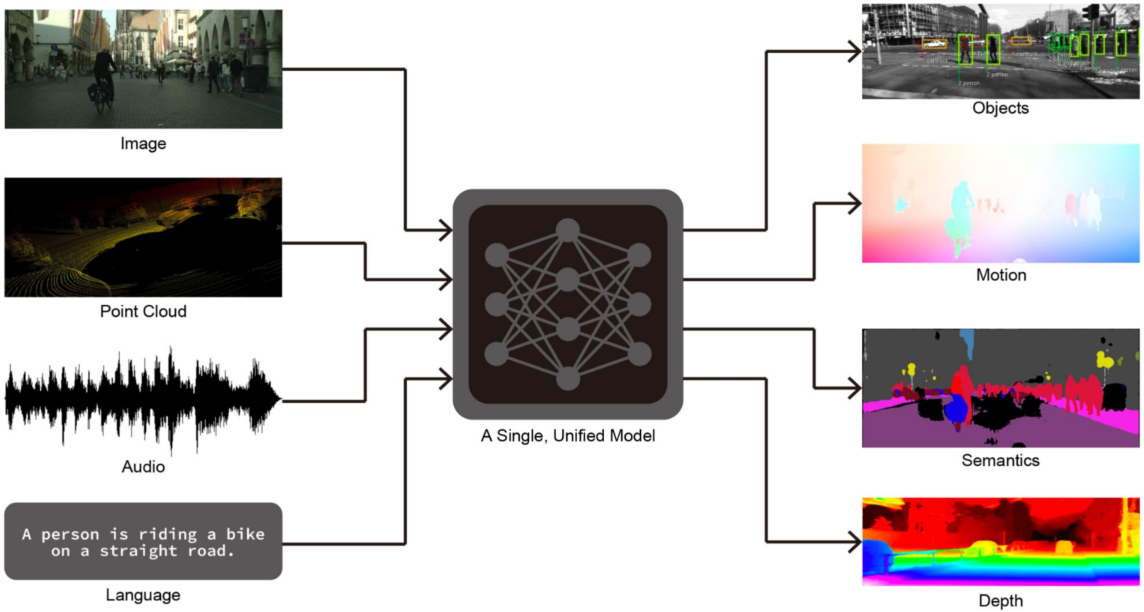
- Many neural network architectures rely on certain type of inductive biases
  - Mostly depends on locality and invariance of data
  - Example1: Convolution exploits 2D grid data structure and translation invariance
  - Example 2: RNN exploits 1D chain data structure and first-order Markov assumption

- These assumptions are generally useful to design efficient models

- The problem is that they often not generalize across different data modalities

Towards versatile neural networks

- We want to design a single model that handles many different data / tasks
- What is the most important requirement for this?

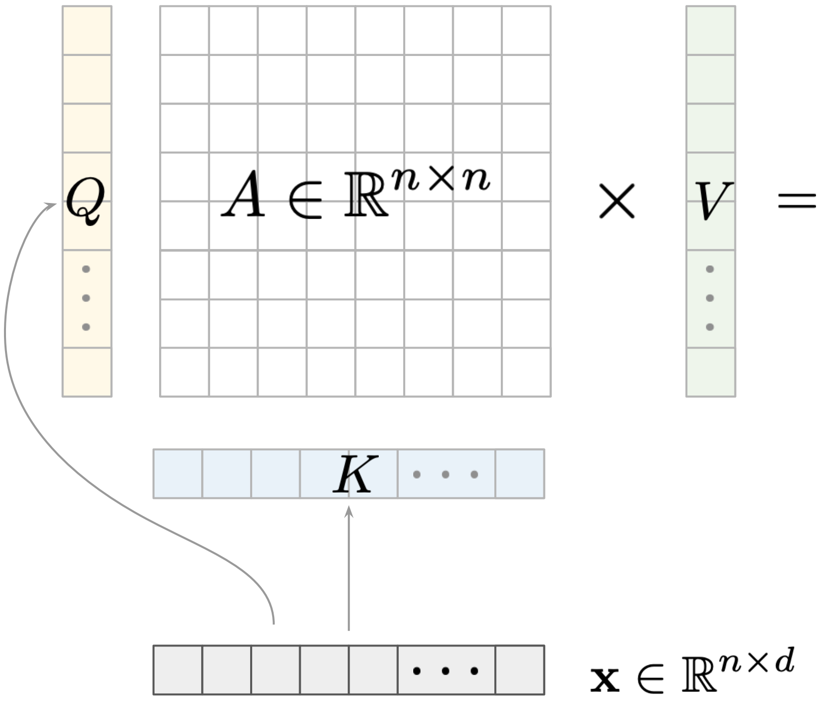


Content

- Revisiting attention and Transformers for versatile architectures
- Extending Transformers to heterogeneous data and tasks

Revisit: Dot-product attention

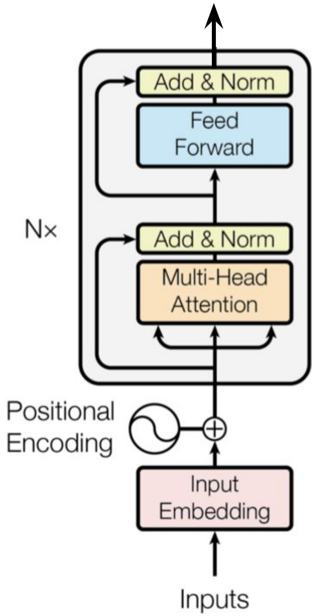
$$\mathrm{Attn}(\mathbf{x}W_q, \mathbf{x}W_k, \mathbf{x}W_v) = \mathrm{softmax}\left(\frac{\mathbf{x}W_q(\mathbf{x}W_k)^T}{\sqrt{d}}\right)\mathbf{x}W_v$$



The output is contexture encoding of input (receptive field=the whole)

Revisit: Transformers (encoder only)

- Input is tokenized $\mathbf{x} \in \mathbb{R}^{n \times d}$



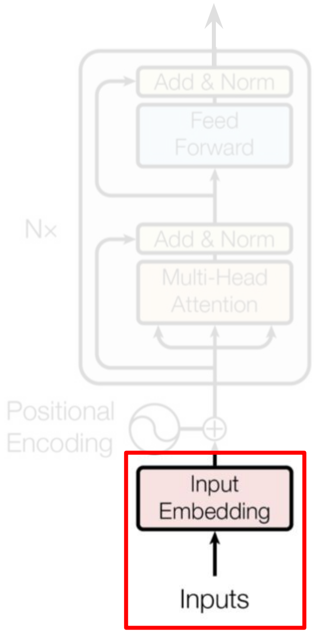
Vaswani et al., Attention Is All You Need, In NeurIPS, 2017

Revisit: Transformers (encoder only)

- Input is tokenized $\mathbf{x} \in \mathbb{R}^{n \times d}$

- Each token is encoded into continuous embedding

$$\mathbf{h}_i = \mathrm{FF}(\mathbf{x}_i)$$



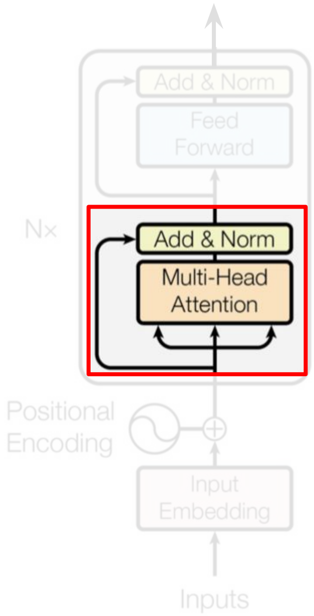
Vaswani et al., Attention Is All You Need, In NeurIPS, 2017

Revisit: Transformers (encoder only)

- Input is tokenized $\mathbf{x} \in \mathbb{R}^{n \times d}$

- Each token is encoded into continuous embedding
$$\mathbf{h}_i = \mathrm{FF}(\mathbf{x}_i)$$

- The tokens are then applied contexture encoding based on multi-head attention with skip connection
$$\mathbf{h}_i = \mathrm{LayerNorm}(\mathbf{h}_i + \mathrm{Attn}(\mathbf{h}_i, \mathbf{h}, \mathbf{h}))$$



Vaswani et al., Attention Is All You Need, In NeurIPS, 2017

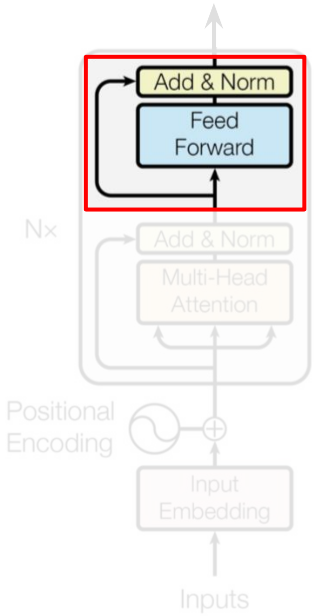
Revisit: Transformers (encoder only)

- Input is tokenized $\mathbf{x} \in \mathbb{R}^{n \times d}$

- Each token is encoded into continuous embedding
  $$\mathbf{h}_i = \mathrm{FF}(\mathbf{x}_i)$$

- The tokens are then applied contexture encoding based on multi-head attention with skip connection
  $$\mathbf{h}_i = \mathrm{LayerNorm}(\mathbf{h}_i + \mathrm{Attn}(\mathbf{h}_i, \mathbf{h}, \mathbf{h}))$$

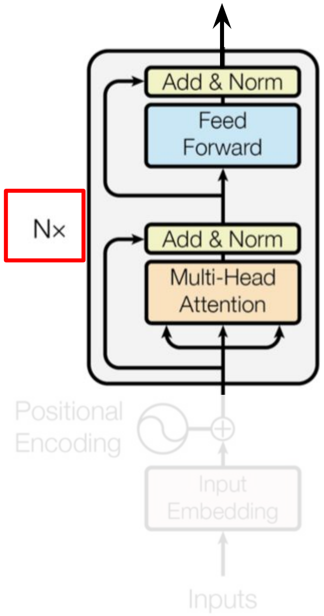
- It applies additional encode with skip connection
  $$\mathbf{h}_i = \mathrm{LayerNorm}(\mathbf{h}_i + \mathrm{FF}(\mathbf{h}_i))$$



Vaswani et al., Attention Is All You Need, In NeurIPS, 2017

Revisit: Transformers (encoder only)

- Input is tokenized $\mathbf{x} \in \mathbb{R}^{n \times d}$
- Each token is encoded into continuous embedding
  $\mathbf{h}_i = \mathrm{FF}(\mathbf{x}_i)$
- The tokens are then applied contexture encoding based on multi-head attention with skip connection
  $\mathbf{h}_i = \mathrm{LayerNorm}(\mathbf{h}_i + \mathrm{Attn}(\mathbf{h}_i, \mathbf{h}, \mathbf{h}))$
- It applies additional encode with skip connection
  $\mathbf{h}_i = \mathrm{LayerNorm}(\mathbf{h}_i + \mathrm{FF}(\mathbf{h}_i))$
- The output is recursively served as the next input



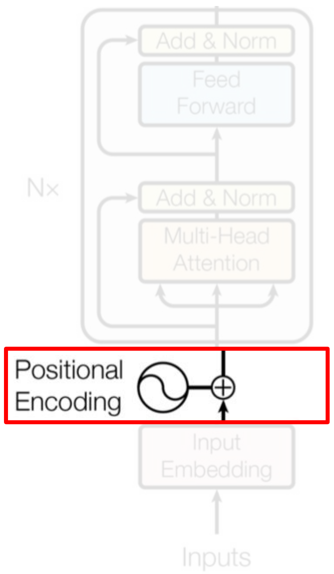
Vaswani et al., Attention Is All You Need, In NeurIPS, 2017

Revisit: Transformers (encoder only)

- When encoding tokens,
  we may add extra embedding that encodes
  position of the token.

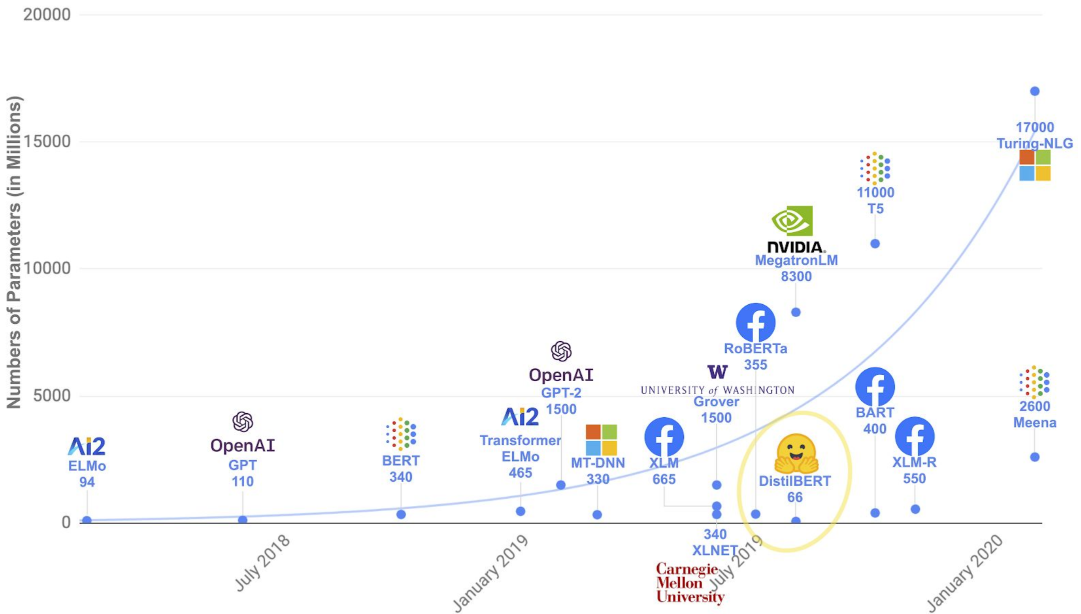
- We can use any periodic function for positional embedding, or even learn how to encode position

$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{\text{model}}})$$
$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{\text{model}}})$$



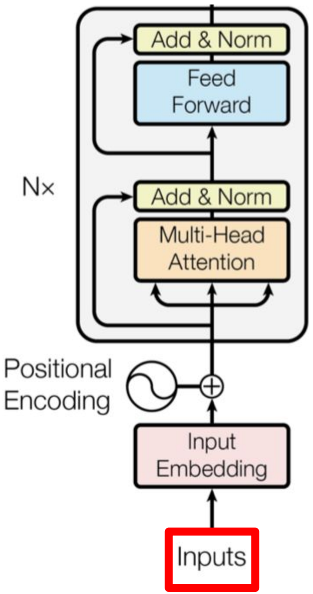
Transformer: performance

- The original Transformer paper was developed for machine translation
- It turned out that Transformer scales surprisingly well with large-scale dataset
- Most innovations in large-scale models are based on Transformer architecture



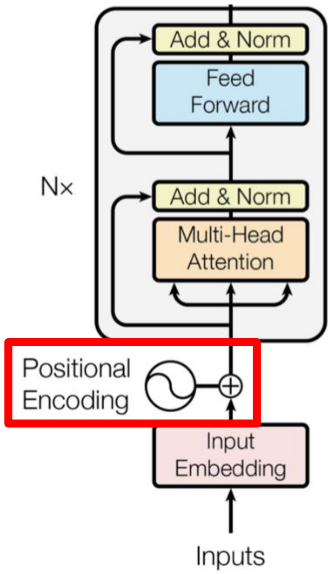
Transformer: analysis

- What are the inductive biases in the Transformers?
  - Data can be represented as a set of "tokens"



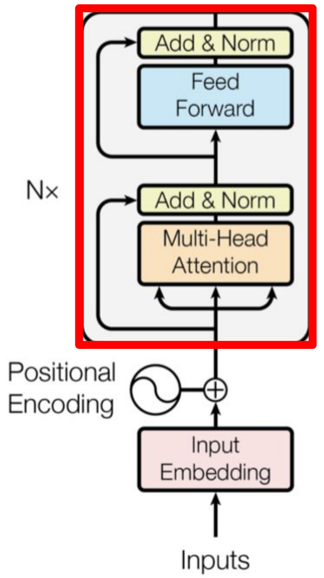
Transformer: analysis

- What are the inductive biases in the Transformers?
  - Data can be represented as a set of “tokens”
  - Position of the tokens can be represented by the “positional encoding”

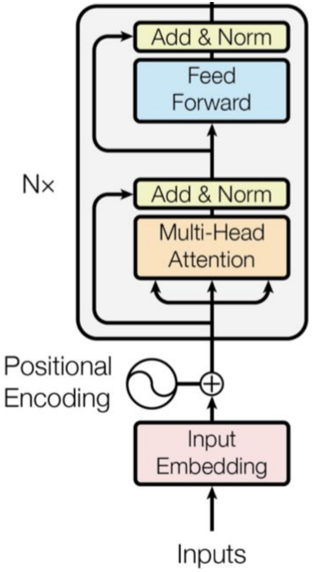


Transformer: analysis

- What are the inductive biases in the Transformers?
  - Data can be represented as a set of “tokens”
  - Position of the tokens can be represented by the “positional encoding”
  - Patterns in the data can be discovered by the “pairwise relational reasoning”



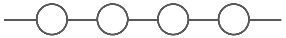
Transformer: analysis



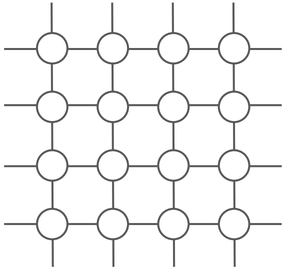
- What are the inductive biases in the Transformers?
  - Data can be represented as a set of “tokens”
  - Position of the tokens can be represented by the “positional encoding”
  - Patterns in the data can be discovered by the “pairwise relational reasoning

- Are these inductive biases generally applicable across data?

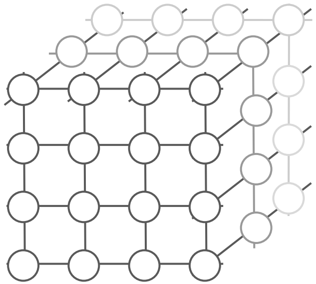
Language



Image

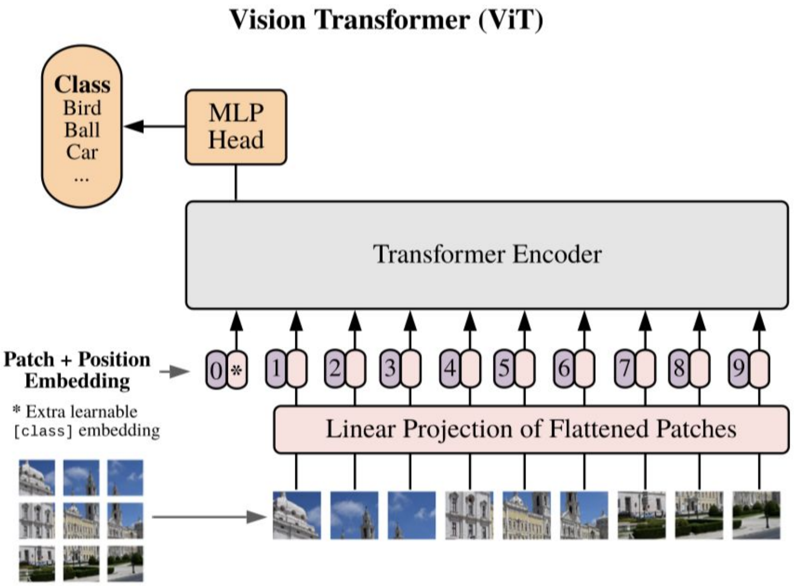


Video



Transformers for vision

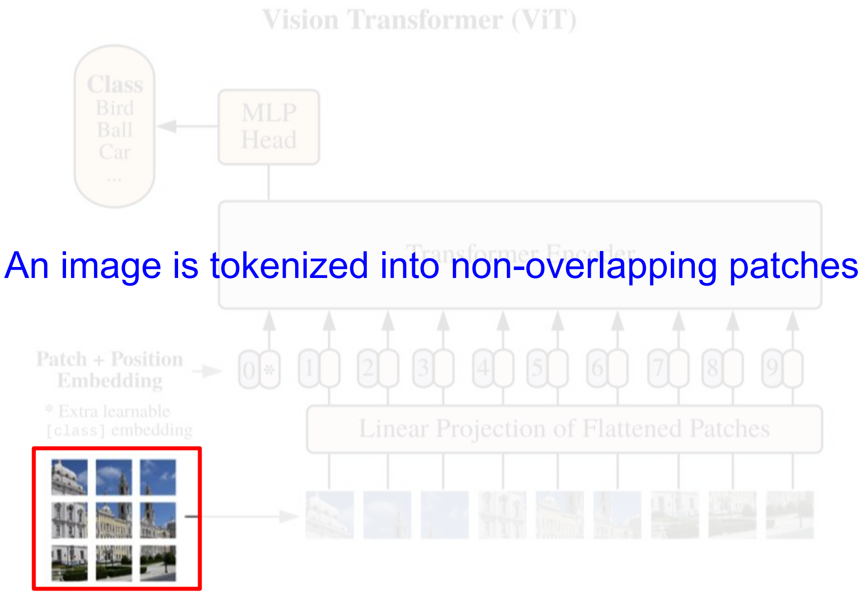
- Vision Transformer for image classification



Dosovitskiy et al., An Image Is Worth 16 X 16 Words: Transformers For Image Recognition At Scale, In NeurIPS, 2017

Transformers for vision

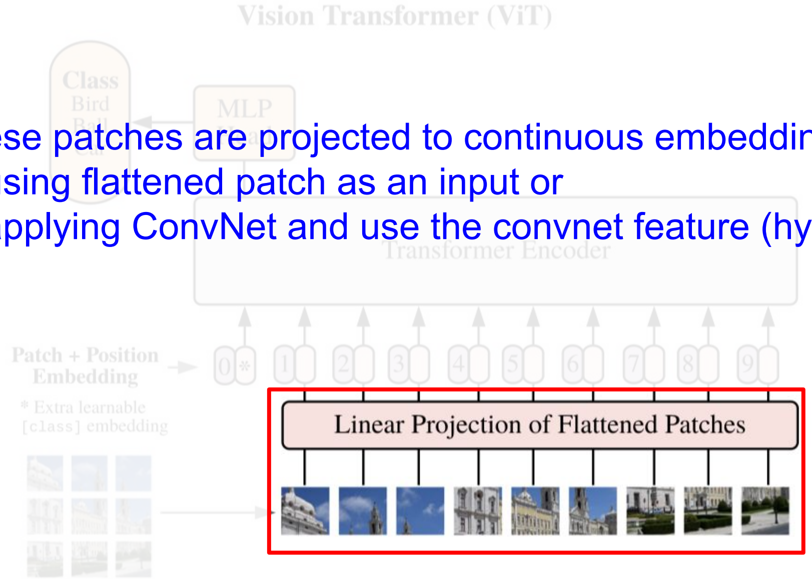
- Vision Transformer for image classification



Dosovitskiy et al., An Image Is Worth 16 X 16 Words: Transformers For Image Recognition At Scale, In NeurIPS, 2017

Transformers for vision

- Vision Transformer for image classification

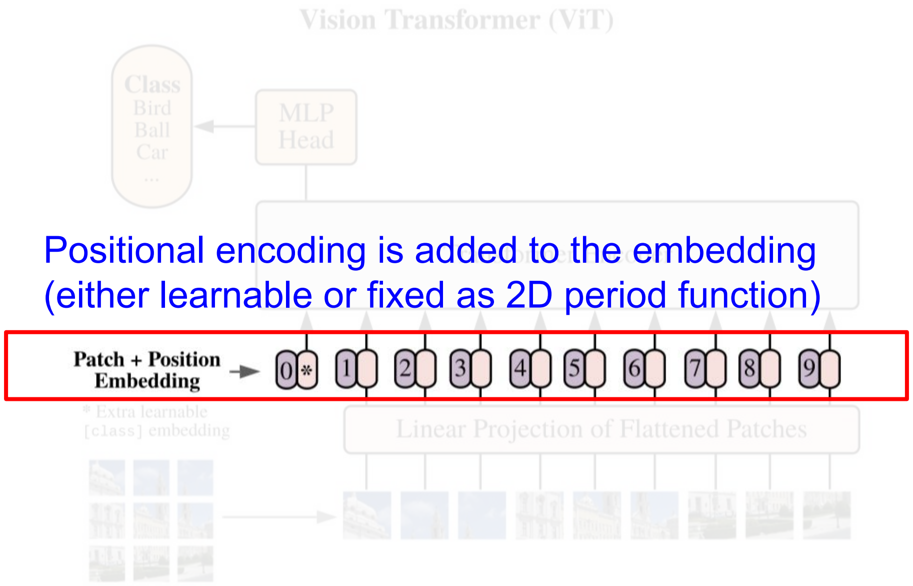


These patches are projected to continuous embedding by
1) using flattened patch as an input or
2) applying ConvNet and use the convnet feature (hybrid)

Dosovitskiy et al., An Image Is Worth 16 X 16 Words: Transformers For Image Recognition At Scale, In NeurIPS, 2017

Transformers for vision

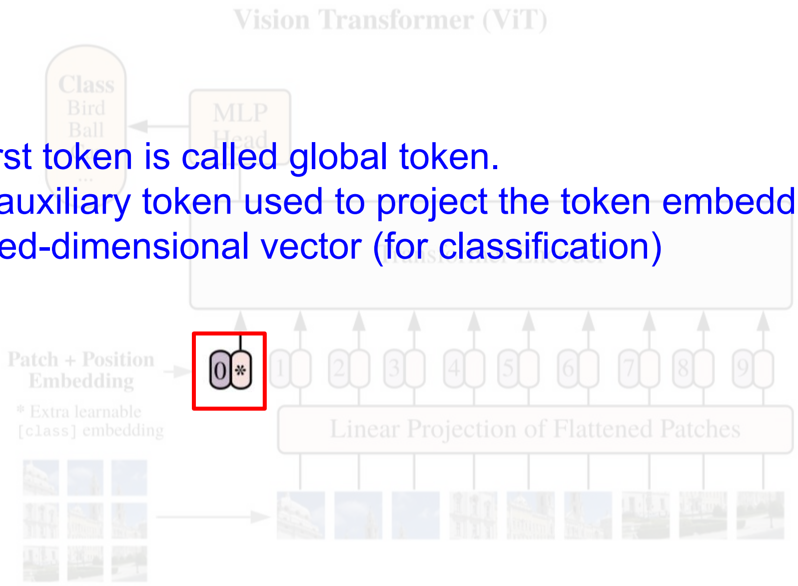
- Vision Transformer for image classification



Dosovitskiy et al., An Image Is Worth 16 X 16 Words: Transformers For Image Recognition At Scale, In NeurIPS, 2017

Transformers for vision

- Vision Transformer for image classification

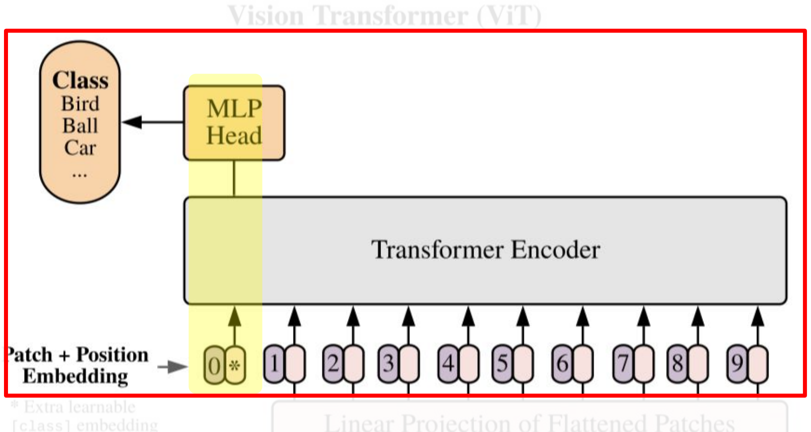


The first token is called global token.
It it is auxiliary token used to project the token embeddings into the fixed-dimensional vector (for classification)

Dosovitskiy et al., An Image Is Worth 16 X 16 Words: Transformers For Image Recognition At Scale, In NeurIPS, 2017

Transformers for vision

- Vision Transformer for image classification

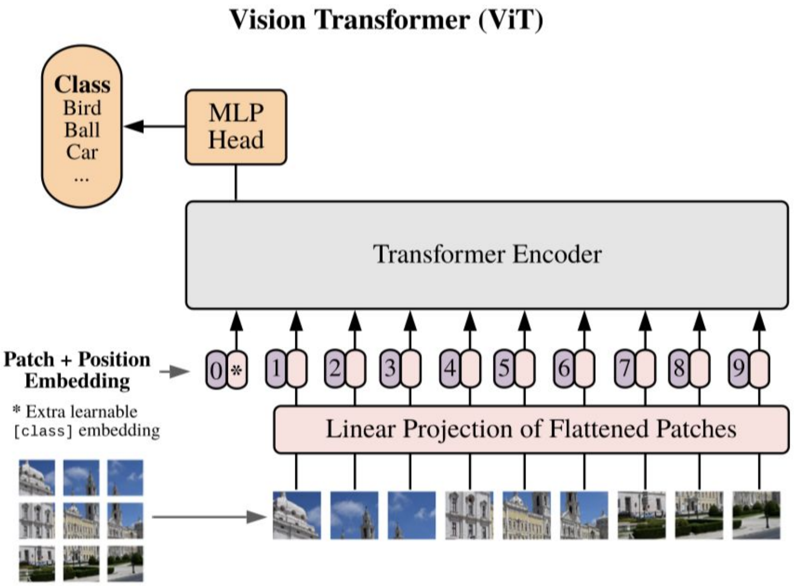


The tokens are processed by Transformer, and the classifier head is applied to the global token to produce class prediction.

Dosovitskiy et al., An Image Is Worth 16 X 16 Words: Transformers For Image Recognition At Scale, In NeurIPS, 2017

Transformers for vision

- Vision Transformer for image classification



Dosovitskiy et al., An Image Is Worth 16 X 16 Words: Transformers For Image Recognition At Scale, In NeurIPS, 2017

Transformers for vision

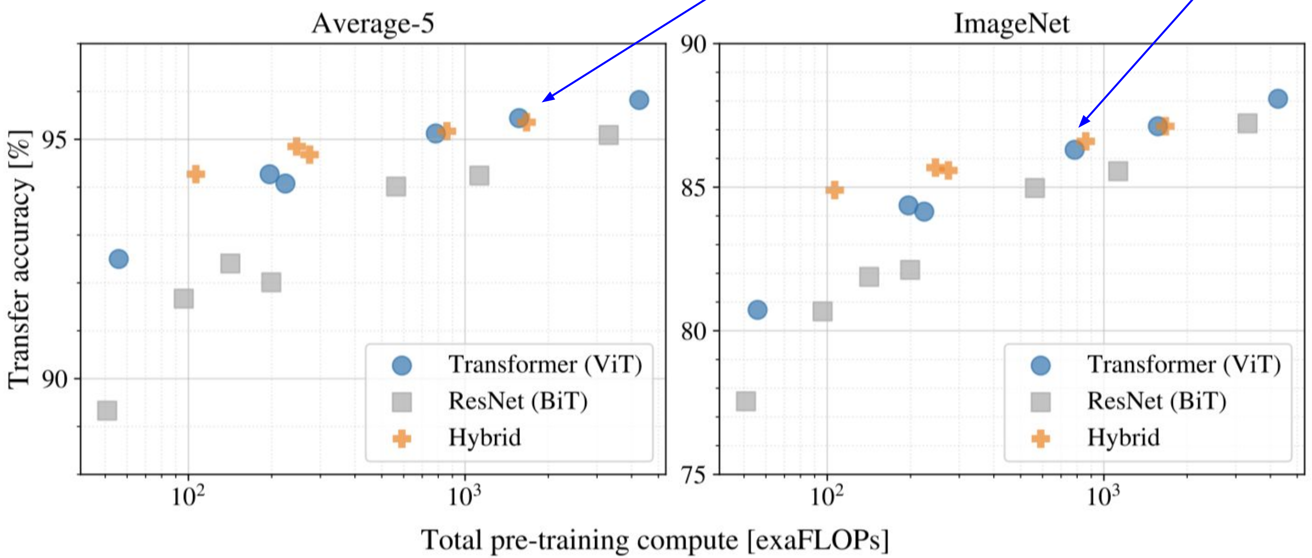
- Vision Transformer outperforms the ResNet with large-scale pre-training

	Ours-JFT (ViT-H/14)	Ours-JFT (ViT-L/16)	Ours-I21k (ViT-L/16)	BiT-L (ResNet152x4)	Noisy Student (EfficientNet-L2)
ImageNet	**88.55** $\pm$ 0.04	87.76 $\pm$ 0.03	85.30 $\pm$ 0.02	87.54 $\pm$ 0.02	88.4/88.5*
ImageNet ReaL	**90.72** $\pm$ 0.05	90.54 $\pm$ 0.03	88.62 $\pm$ 0.05	90.54	90.55
CIFAR-10	**99.50** $\pm$ 0.06	99.42 $\pm$ 0.03	99.15 $\pm$ 0.03	99.37 $\pm$ 0.06	–
CIFAR-100	**94.55** $\pm$ 0.04	93.90 $\pm$ 0.05	93.25 $\pm$ 0.05	93.51 $\pm$ 0.08	–
Oxford-IIIT Pets	**97.56** $\pm$ 0.03	97.32 $\pm$ 0.11	94.67 $\pm$ 0.15	96.62 $\pm$ 0.23	–
Oxford Flowers-102	99.68 $\pm$ 0.02	**99.74** $\pm$ 0.00	99.61 $\pm$ 0.02	99.63 $\pm$ 0.03	–
VTAB (19 tasks)	**77.63** $\pm$ 0.23	76.28 $\pm$ 0.46	72.72 $\pm$ 0.21	76.29 $\pm$ 1.70	–
TPUv3-core-days	2.5k	0.68k	0.23k	9.9k	12.3k

Transformers for vision

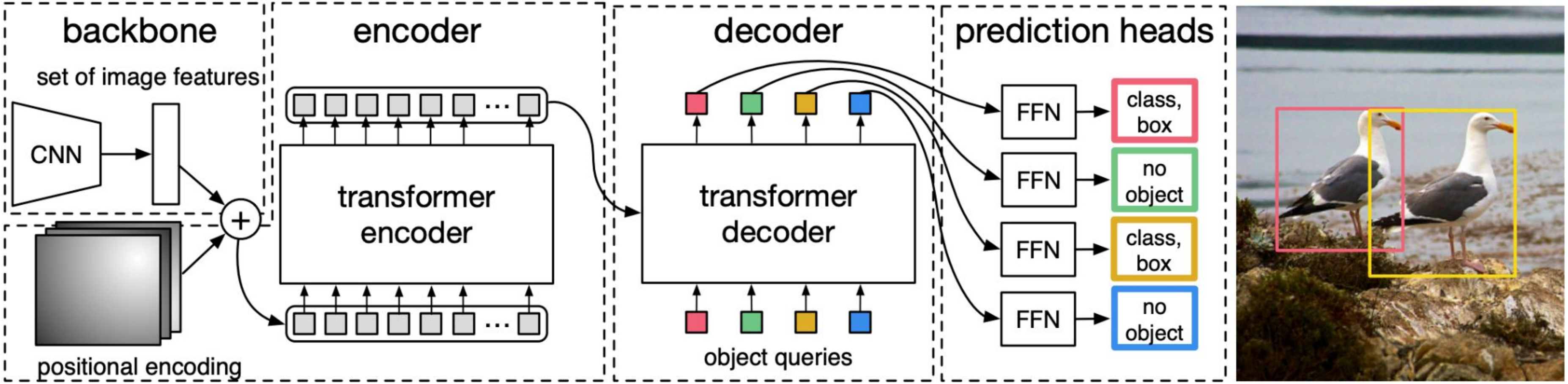
- ViT scales well with the larger data

The benefit from hybrid architecture vanishes as it is learned from larger data



Transformers for structured vision prediction

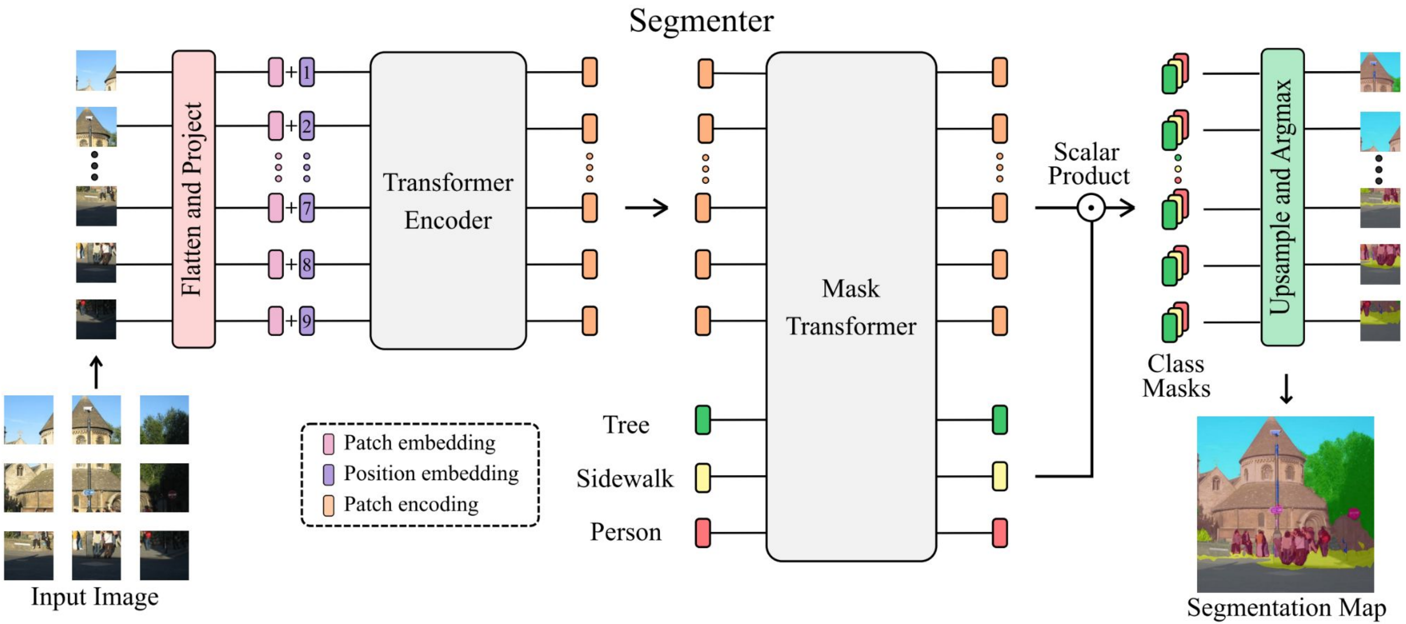
- Transformer + object detection



Carion et al., End-to-End Object Detection with Transformers, In ECCV, 2020

Transformers for structured vision prediction

- Transformer + semantic segmentation



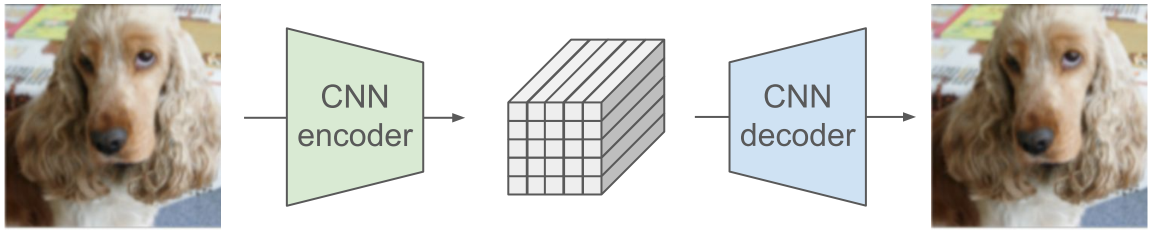
Strudel et al., Segmenter: Transformer for Semantic Segmentation, In ICCV, 2021

Can we treat images as words?

- Vision Transformers (ViT) treat patches as tokens
- However, these patches are continuous while words are discrete
- If we can discretize the images (patches) as words,
  then there is no difference in language and vision

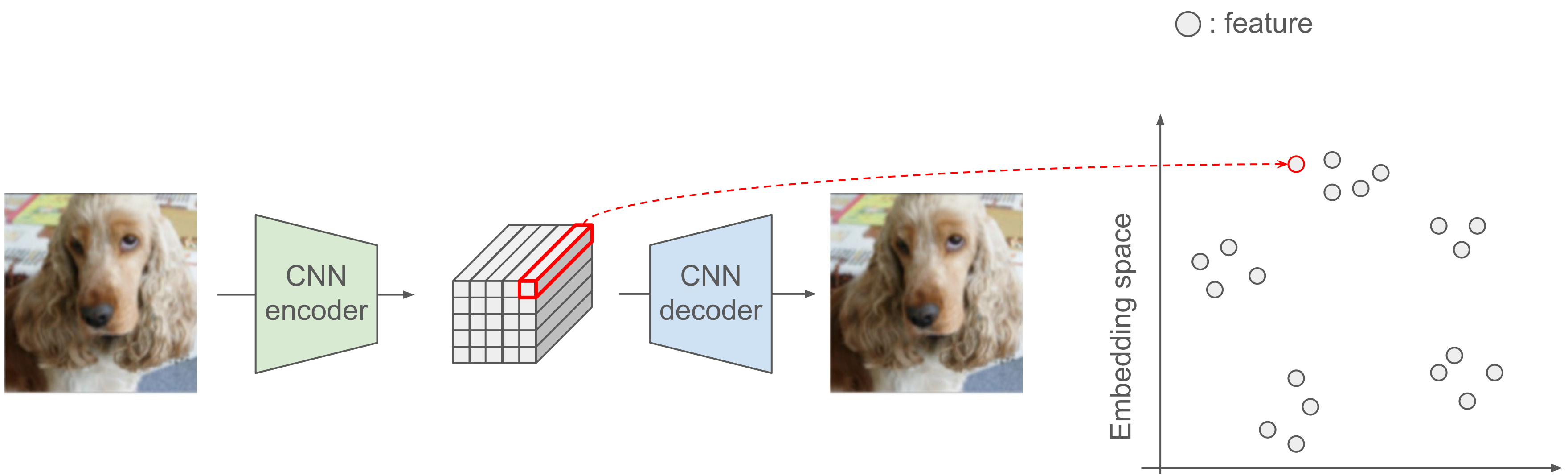
Discretizing image representation

- Basic idea
  - Learn an autoencoder (or VAE)



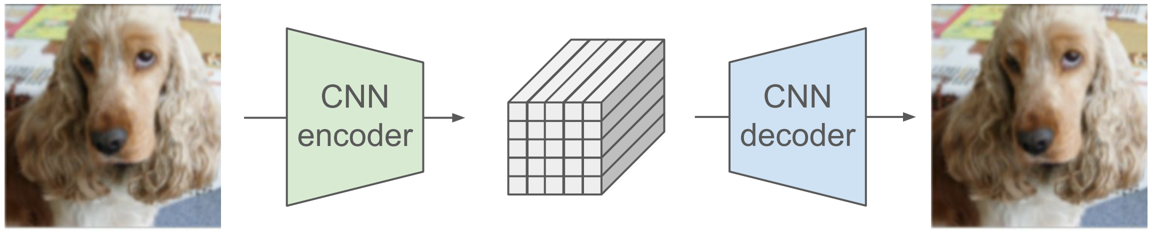
Discretizing image representation

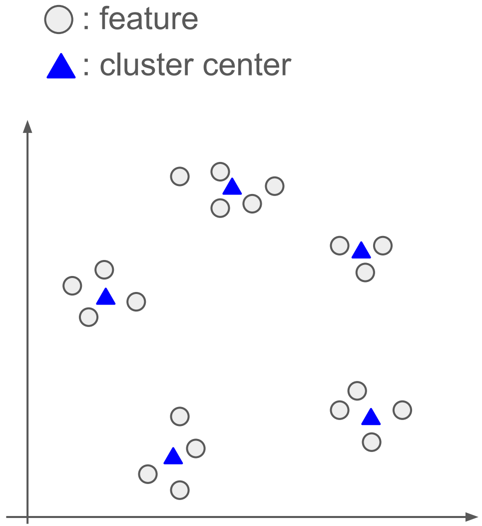
- Basic idea
  - Learn an autoencoder (or VAE)
  - Collect all learned (patch) embeddings of training images



Discretizing image representation

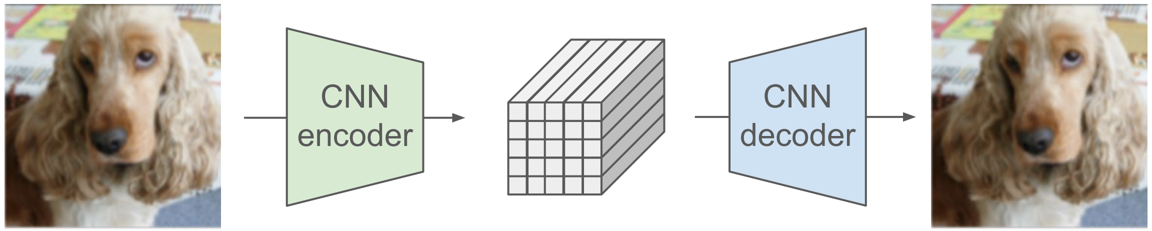
- Basic idea
  - Learn an autoencoder (or VAE)
  - Collect all features of training images
  - Then apply the clustering in the embedding space, which will give us cluster centers

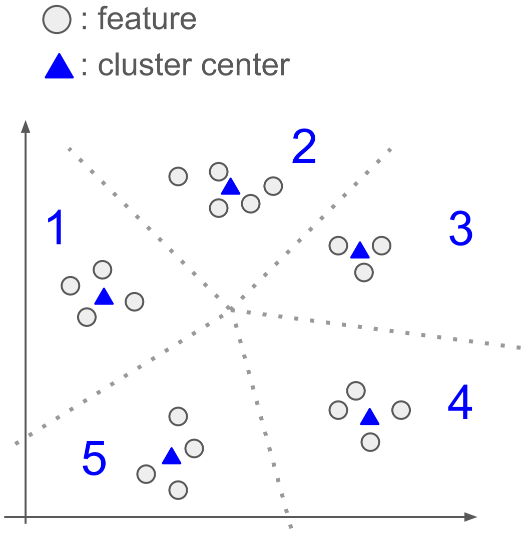




Discretizing image representation

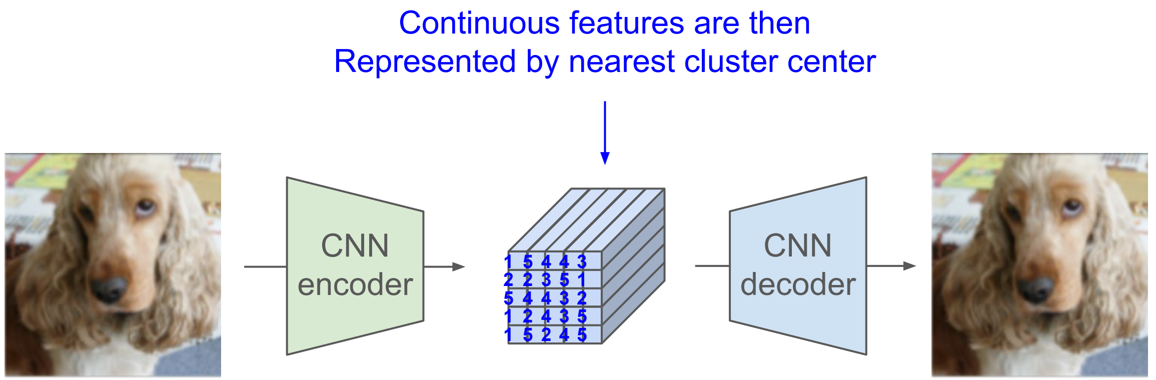
- Basic idea
  - Learn an autoencoder (or VAE)
  - Collect all features of training images
  - Then apply the clustering in the embedding space, which will give us cluster centers
  - Assigning the unique indices to the cluster centers, every continuous embeddings can be assigned with discrete index by associating them to nearest cluster centers

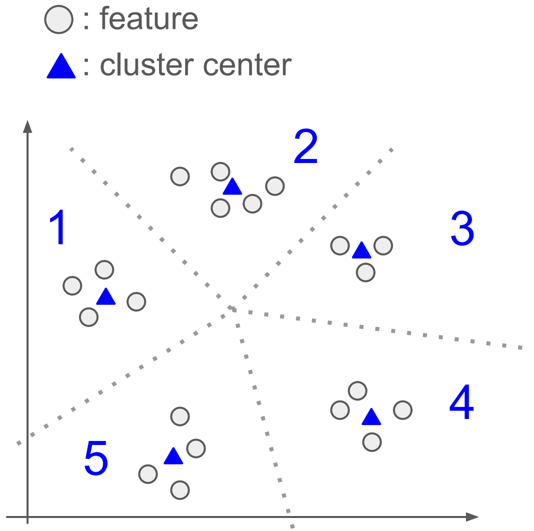




Discretizing image representation

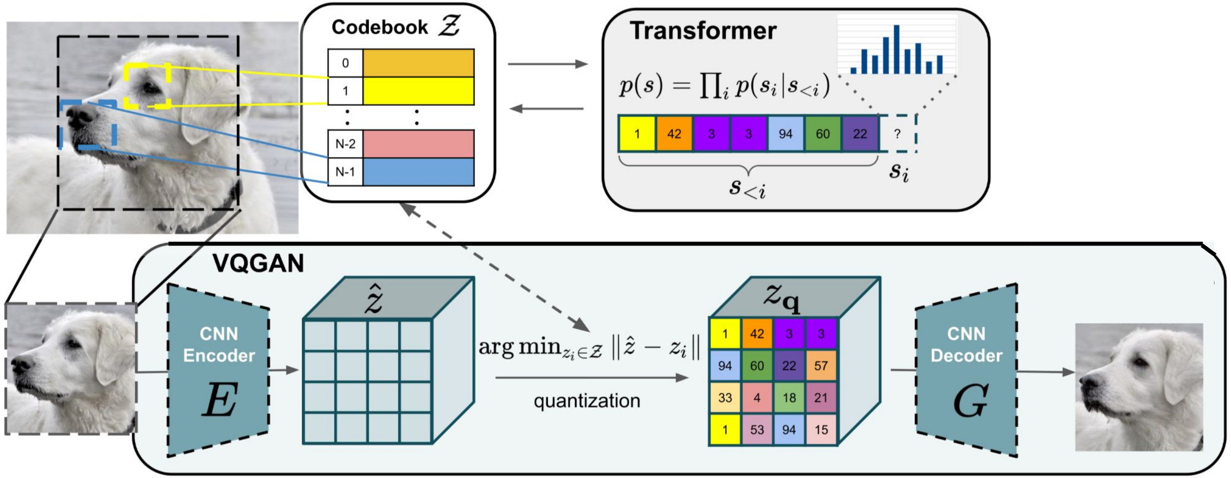
- Basic idea
  - Learn an autoencoder (or VAE)
  - Collect all features of training images
  - Then apply the clustering in the embedding space, which will give us cluster centers
  - Assigning the unique indices to the cluster centers, every continuous embeddings can be assigned with discrete index by associating them to nearest cluster centers





Vector quantization

- Learn to cluster (quantize) the features end-to-end with autoencoding



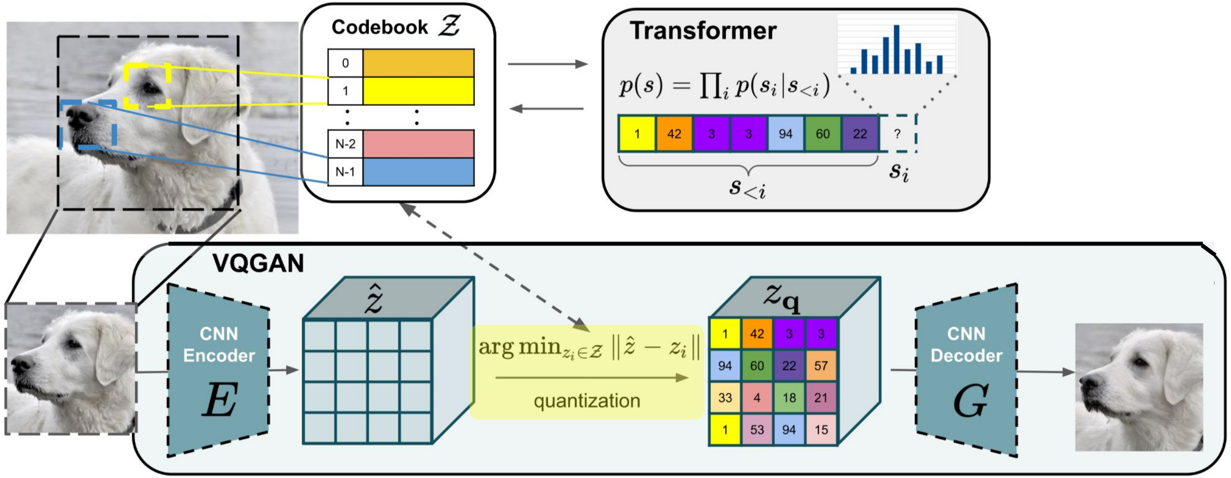
Esser et al., Taming Transformers for High-Resolution Image Synthesis, 2021

Vector quantization

- Learn to cluster (quantize) the features end-to-end with autoencoding

$$z_{\mathbf{q}} = \mathbf{q}(\hat{z}) := \left(\underset{z_k \in \mathcal{Z}}{\arg\min} \|\hat{z}_{ij} - z_k\| \right) \in \mathbb{R}^{h \times w \times n_z}$$

Find the nearest codebook (cluster center) and replace the feature with it



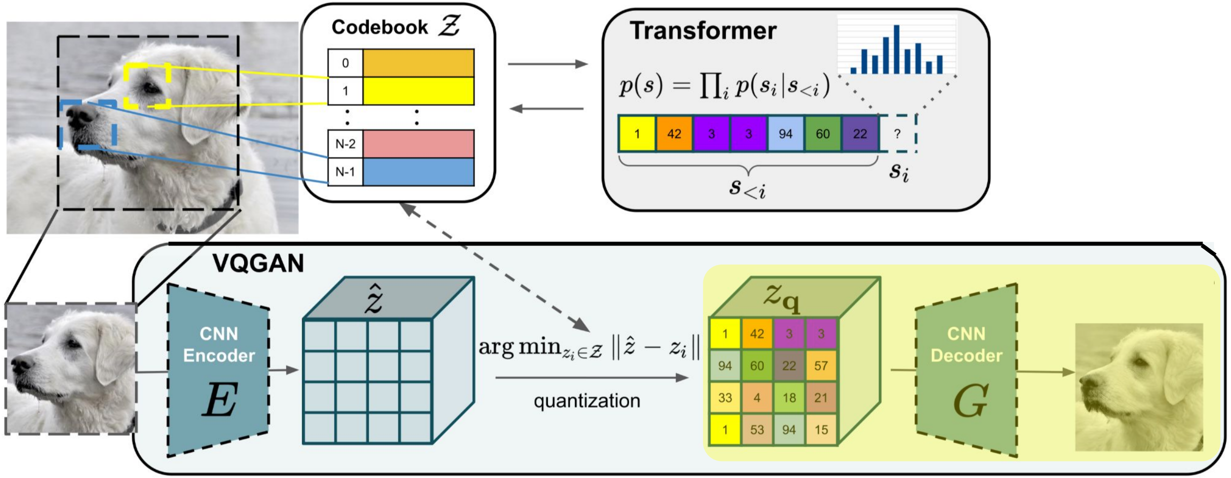
Esser et al., Taming Transformers for High-Resolution Image Synthesis, 2021

Vector quantization

- Learn to cluster (quantize) the features end-to-end with autoencoding

$$\hat{x} = G(z_{\mathbf{q}}) = G\left(\mathbf{q}(E(x))\right)$$

The output image is obtained by decoding the quantized features



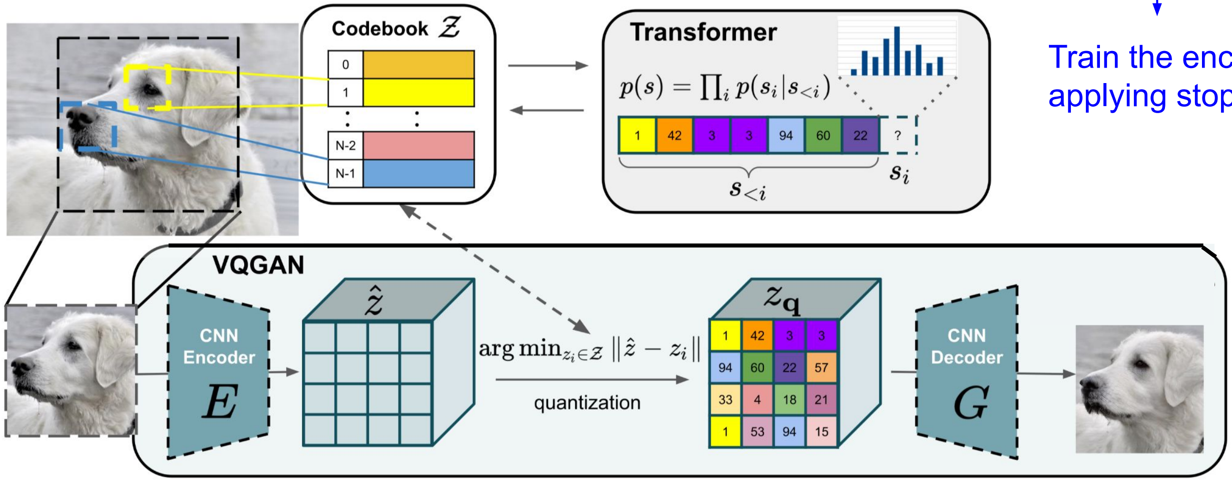
Esser et al., Taming Transformers for High-Resolution Image Synthesis, 2021

Vector quantization

- Learn to cluster (quantize) the features end-to-end with autoencoding

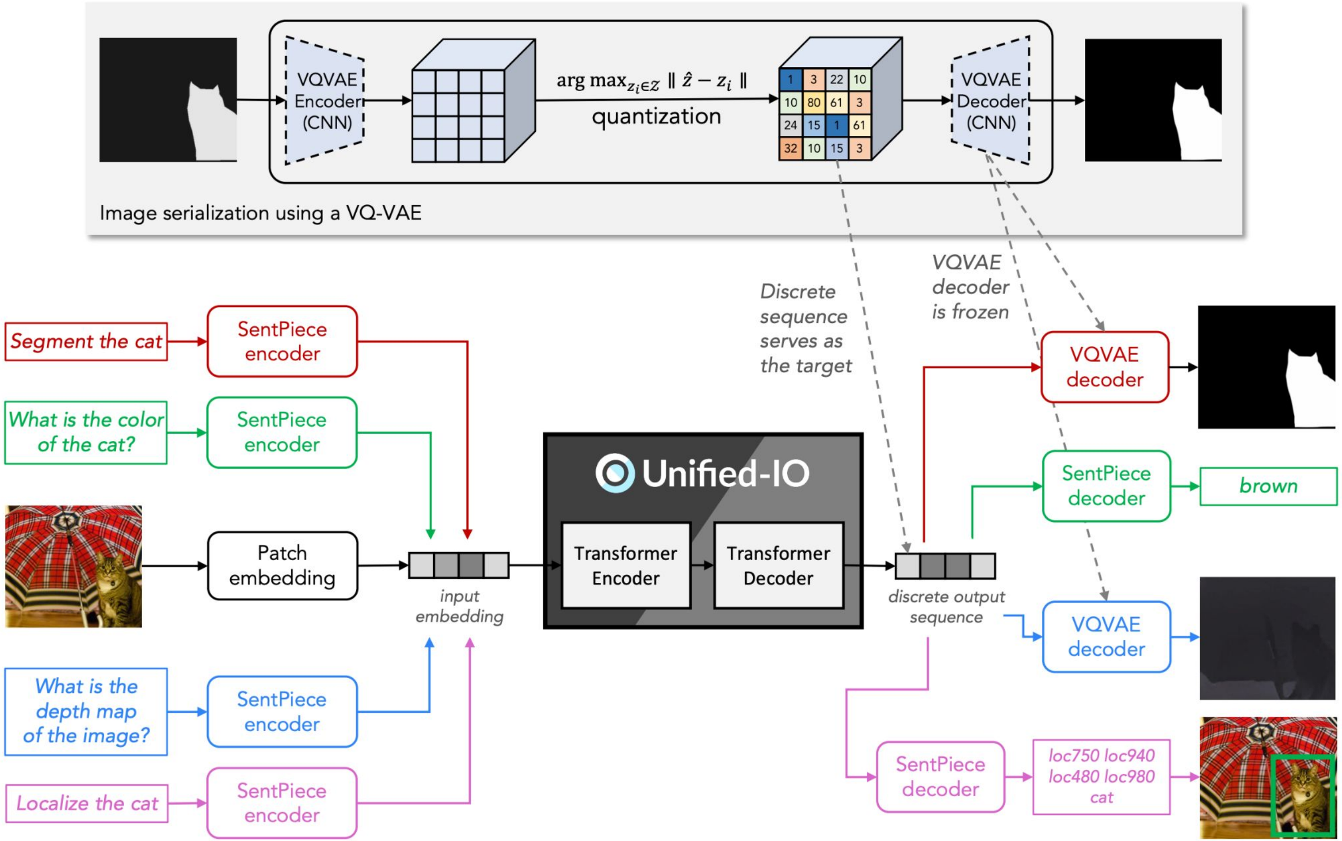
$$\mathcal{L}_{\mathrm{VQ}}(E, G, \mathcal{Z}) = \|x - \hat{x}\|^2 + \|\mathrm{sg}[E(x)] - z_{\mathbf{q}}\|_2^2 + \beta\|\mathrm{sg}[z_{\mathbf{q}}] - E(x)\|_2^2$$

Train the encoder and prototypes by applying stop gradient alternatively



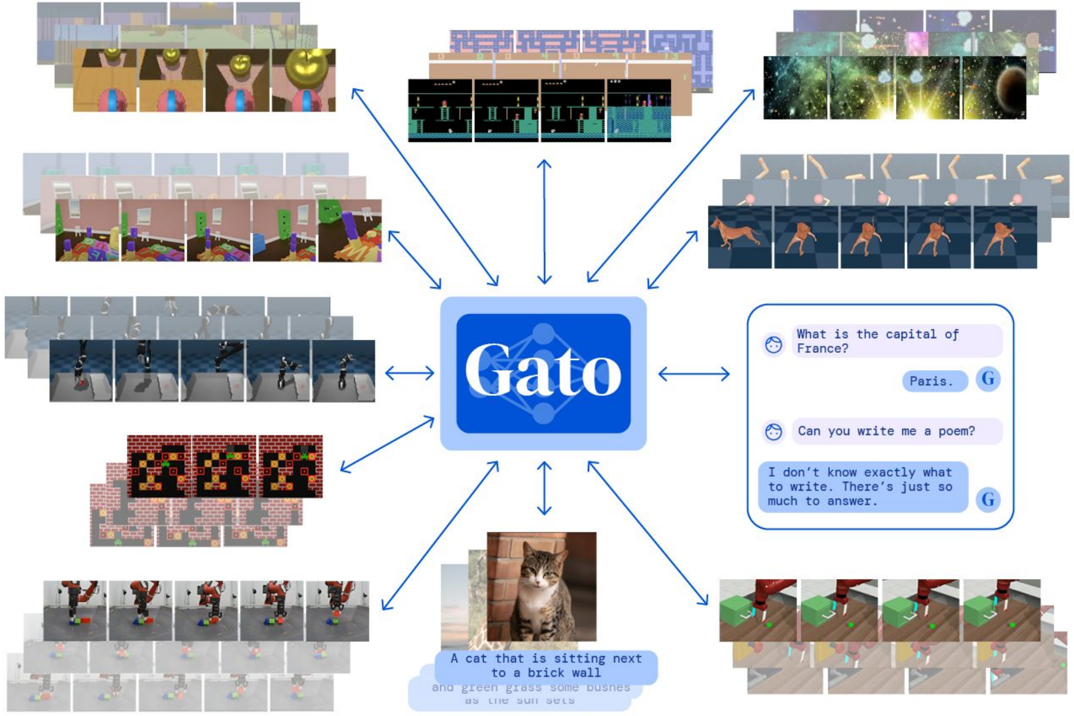
Esser et al., Taming Transformers for High-Resolution Image Synthesis, 2021

Transformers for universal vision learner



Lu et al., Unified-IO: A Unified Model for Vision, Language, and Multi-Modal Tasks, In arXiv, 2022

Transformer for language, vision, and RL agents



Reed et al., A Generalist Agent, In arXiv, 2022