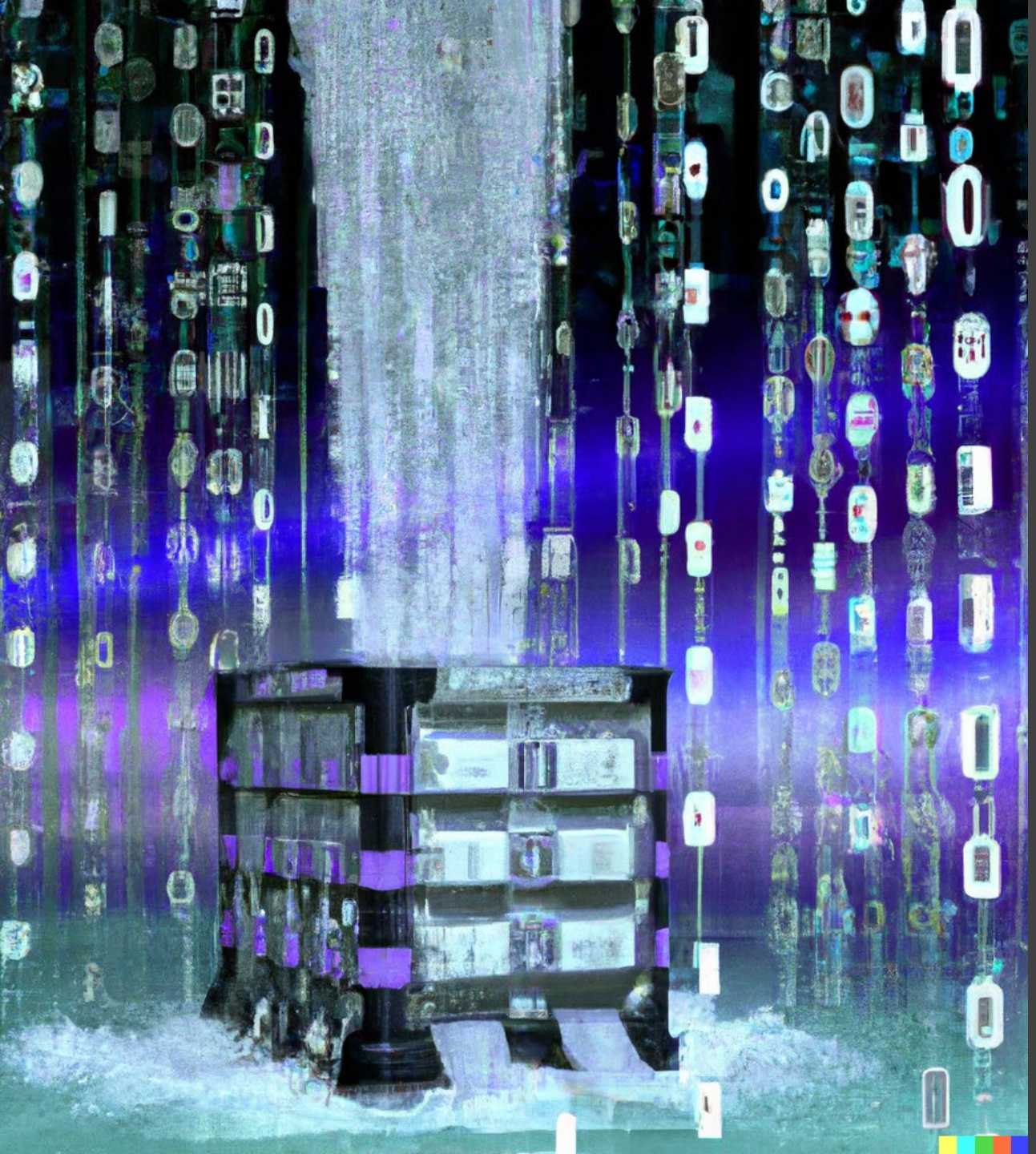


Dall-E

Transformers in Language and Vision

Applied Machine Learning
Derek Hoiem

Self-attention, in simple words

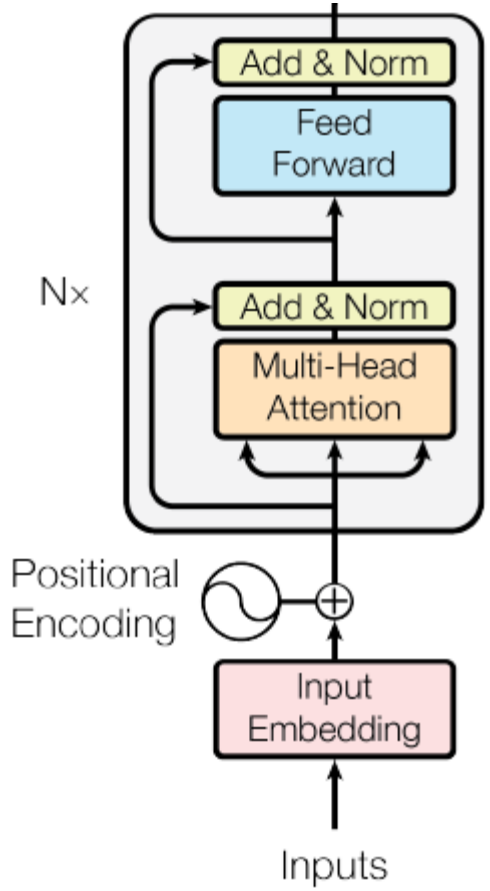
1. Project the input token values into query, key, and value vectors using (linear) weight matrices

2. Using keys and queries, compare each input token to all the tokens (including itself) to get a similarity score

3. Replace the input values with a similarity-weighted sum of all values

Transformers: general data processors

- **Input tokens can represent anything**: image patches, text tokens, audio, controls, etc.
- Invariant to order of tokens: **add positional embedding** to distinguish pos/type of input
- **Transformer block**:
  - Apply multi-head attention
  - Apply 2-layer MLP with ReLU to each token separately
  - Residual and layer norm (over all tokens) after each
- Can stack any number of transformer blocks

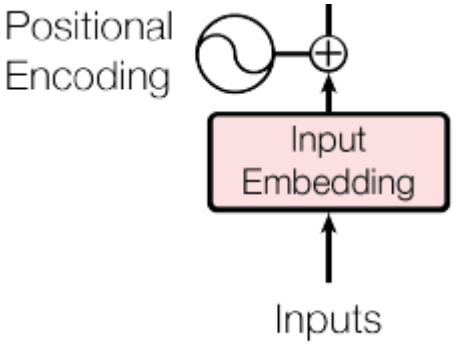


[Attention is all you need]

Positional encodings

- Transformer processing does not depend on position of token
  - This is kind of similar to convolution, as each “patch” or token vector is processed independently, but no overlap between patches
  - But to compare between tokens, relative position may be important
- Sinusoidal encoding (on right) is such that a dot product between encodings corresponds to positional similarity
- Learned or even fixed random encodings also work similarly in practice

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



Remember from last class

Sub-word tokenization based on byte-pair encoding is an effective way to turn natural text into a sequence of integers

Chair is broken →
ch##, ##air, is, brok##, ##en

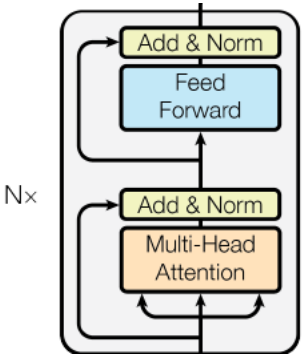
Learned vector embeddings of these integers model the relationships between words

Paris – France
+ Italy = Rome

Attention is a general processing mechanism that regresses or clusters values

Input (k,q,v)	iter 1	iter 2	iter 3	iter 4
1.000	1.497	1.818	1.988	2.147
9.000	8.503	8.182	8.012	7.853
8.000	8.128	8.141	8.010	7.853
2.000	1.872	1.859	1.990	2.147

Stacked transformer blocks are a powerful network architecture that alternates attention and MLPs



Further reading: http://nlp.seas.harvard.edu/annotated-transformer/

Q1

https://tinyurl.com/AML441-L19

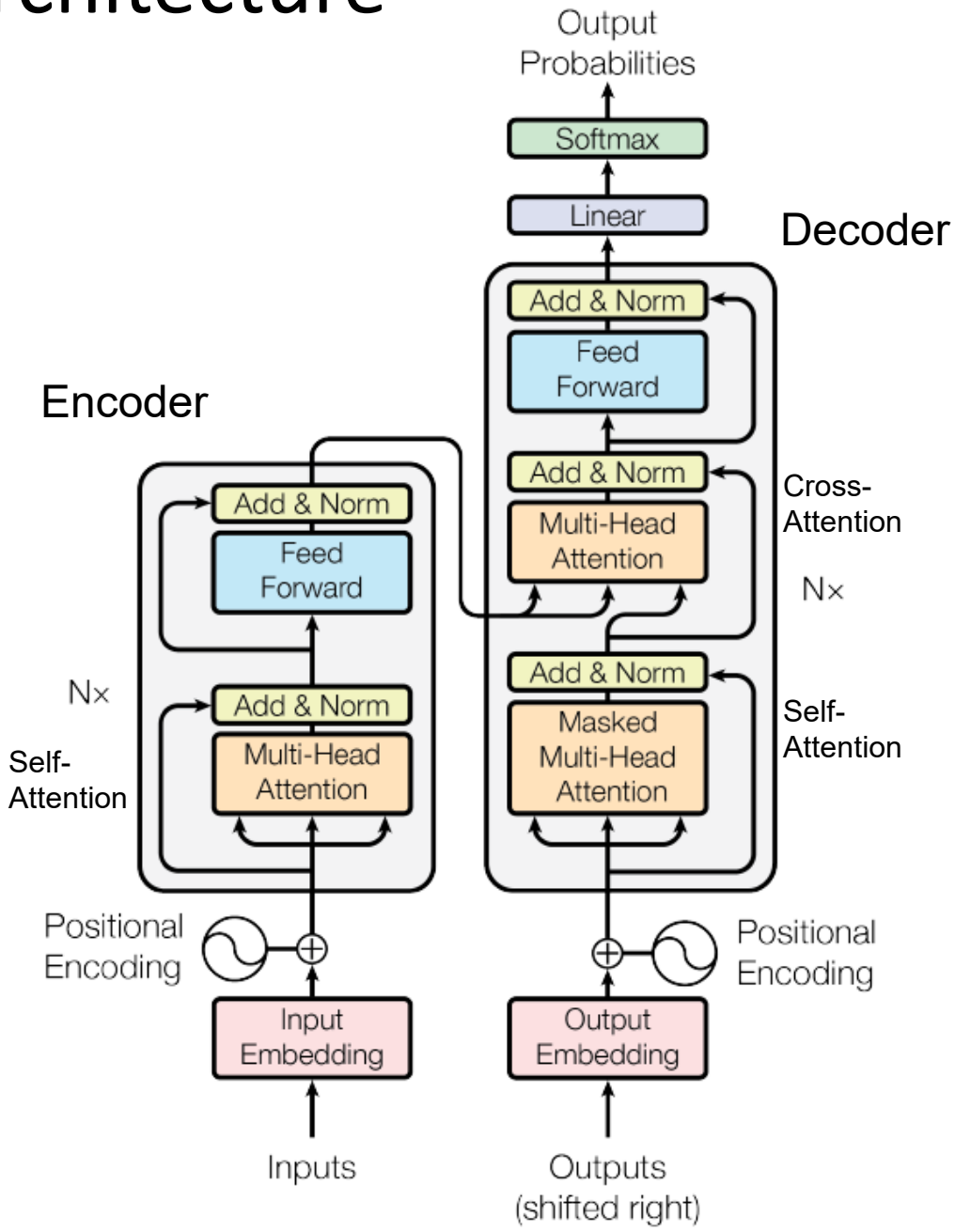


Today’s Lecture

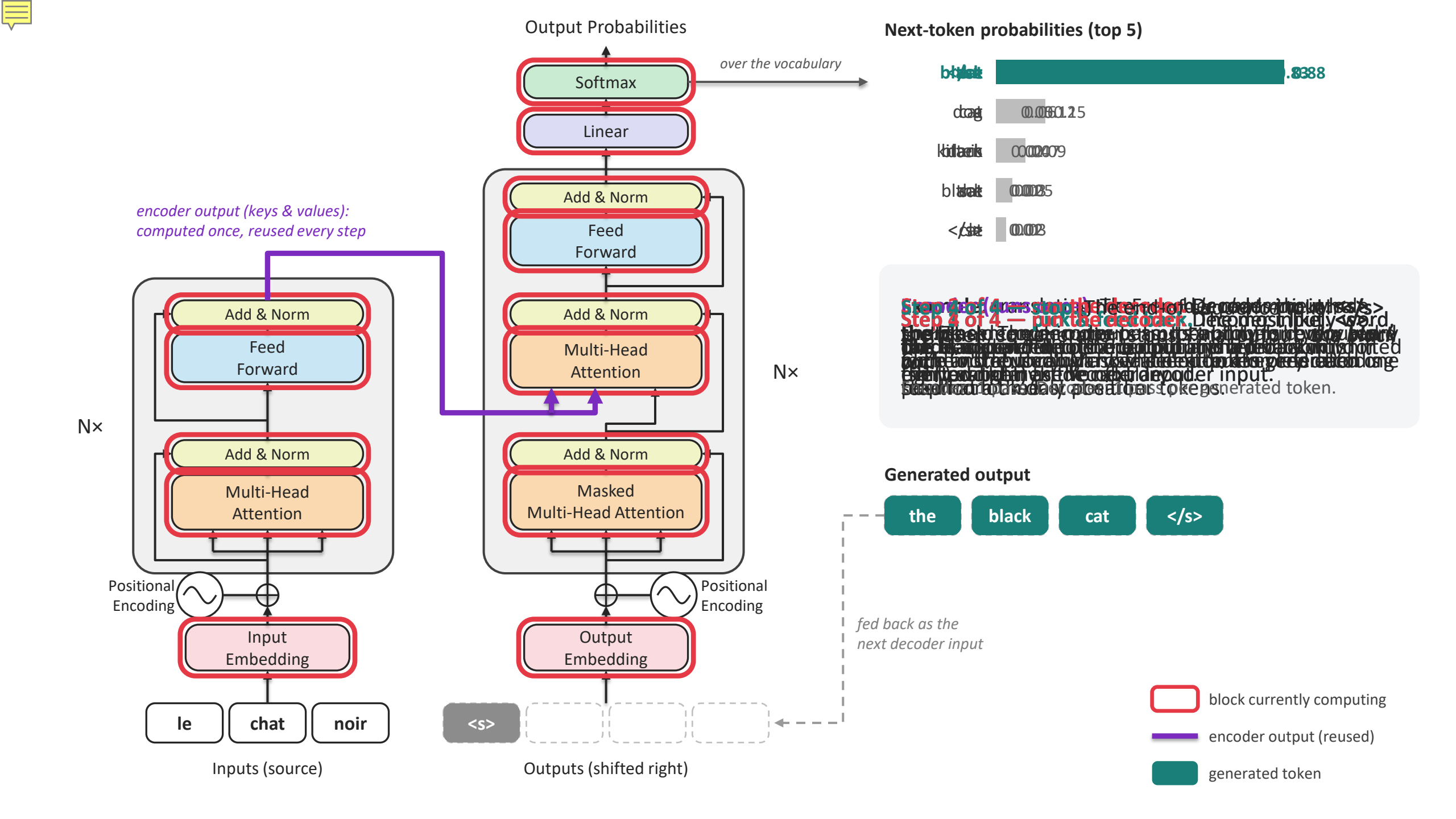
- Language Transformer architecture
- BERT: Large Language Model
- ViT (Vision Transformer): Image classification
- CNN vs Transformer comparison
- Unified-IO: Sequence-to-sequence vision-language

Language Transformer: Complete Architecture

- WordPiece tokens (integers) are mapped to learned 512-d vectors
- Positional encoding added to each vector
- N=6 transformer blocks applied to input
- Until <EOS> is output:
  - Process input + output so far
  - Output most likely word (after more attention blocks and linear model)
- Encoder/decoder architecture for sequence to sequence tasks

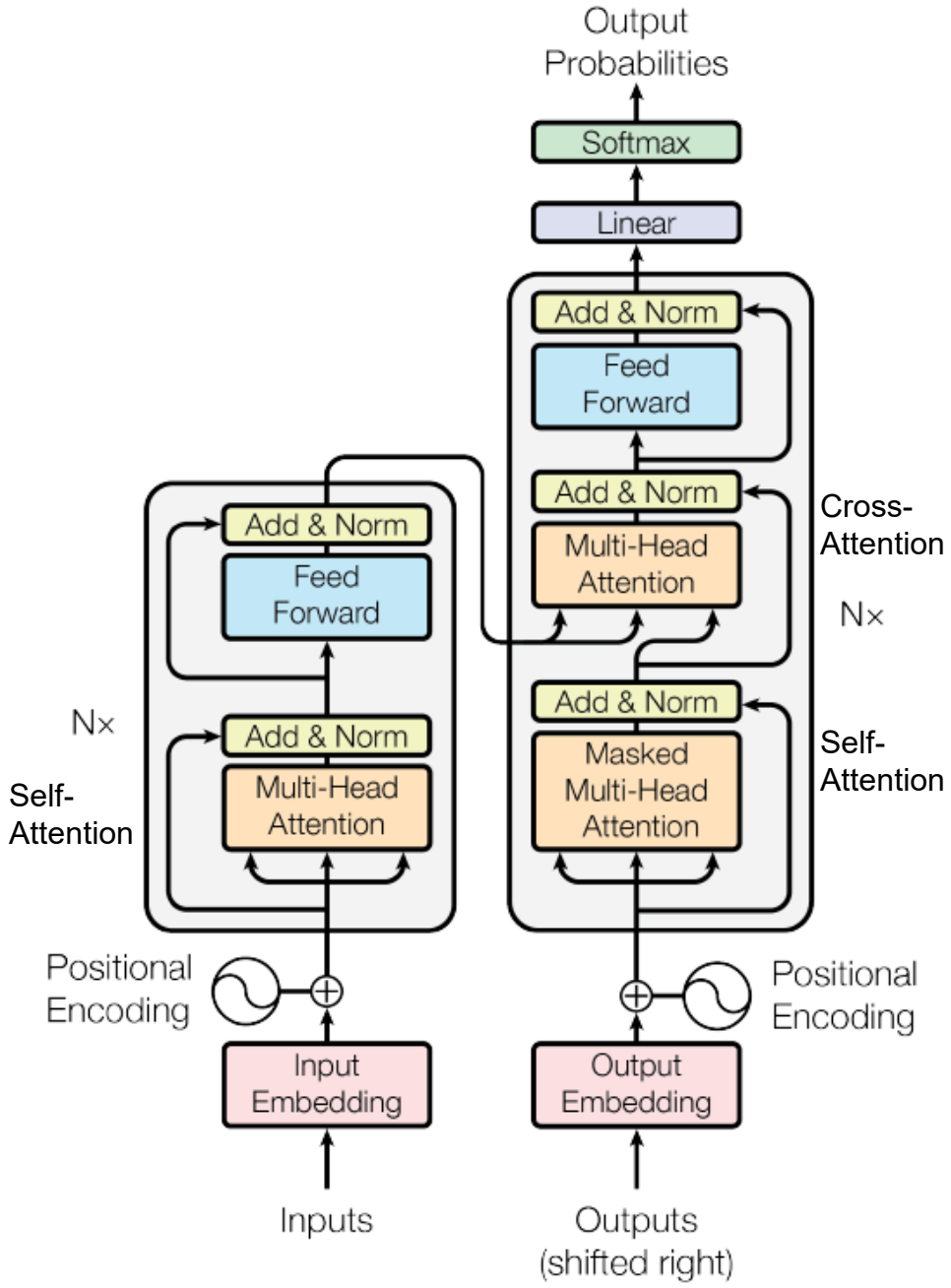


[Attention is all you need]


Output Probabilities
Softmax
over the vocabulary
Linear
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Add & Norm
Masked Multi-Head Attention
N×
encoder output (keys & values):
computed once, reused every step
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Add & Norm
Multi-Head Attention
N×
Positional Encoding
Positional Encoding
Input Embedding
Output Embedding
le
chat
noir
<s>
Inputs (source)
Outputs (shifted right)
Next-token probabilities (top 5)
fed back as the next decoder input
Generated output
the
black
cat
</s>
block currently computing
encoder output (reused)
generated token

Application to Translation

- English-German
  - 4.5M sentence pairs
  - 37K tokens
- English-French
  - 36M sentences
  - 32K tokens
- Base models trained on 8 P100s for 12 hours
- Big models (2x token dim, 3x training steps) trained for 3.5 days
- Adam optimizer: learning rate ramps up for 4K iterations, then down
- Regularization: drop-out, L2 weight, label smoothing



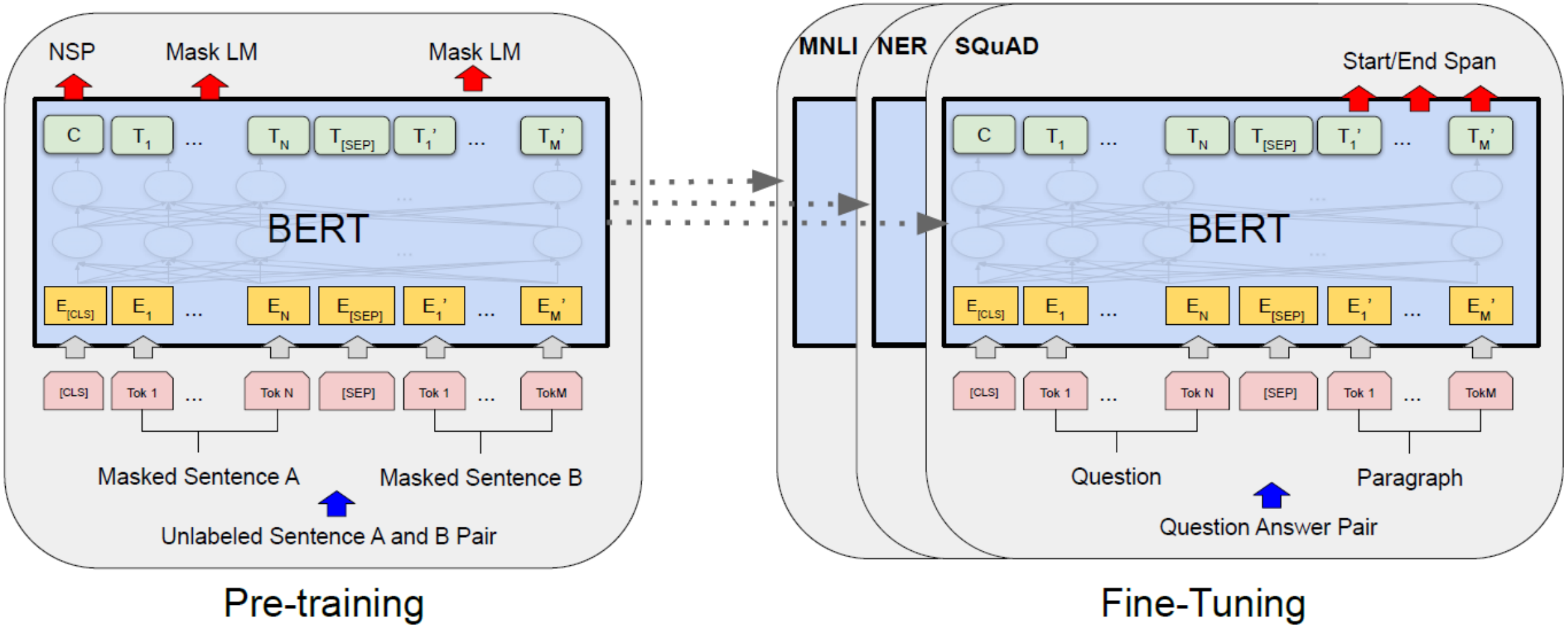
[Attention is all you need]

Results

Table 2: The Transformer achieves better BLEU scores than previous state-of-the-art models on the English-to-German and English-to-French newstest2014 tests at a fraction of the training cost.

Model	BLEU		Training Cost (FLOPs)	
	EN-DE	EN-FR	EN-DE	EN-FR
ByteNet [18]	23.75			
Deep-Att + PosUnk [39]		39.2		$1.0 \cdot 10^{20}$
GNMT + RL [38]	24.6	39.92	$2.3 \cdot 10^{19}$	$1.4 \cdot 10^{20}$
ConvS2S [9]	25.16	40.46	$9.6 \cdot 10^{18}$	$1.5 \cdot 10^{20}$
MoE [32]	26.03	40.56	$2.0 \cdot 10^{19}$	$1.2 \cdot 10^{20}$
Deep-Att + PosUnk Ensemble [39]		40.4		$8.0 \cdot 10^{20}$
GNMT + RL Ensemble [38]	26.30	41.16	$1.8 \cdot 10^{20}$	$1.1 \cdot 10^{21}$
ConvS2S Ensemble [9]	26.36	**41.29**	$7.7 \cdot 10^{19}$	$1.2 \cdot 10^{21}$
Transformer (base model)	27.3	38.1	**$3.3 \cdot 10^{18}$**	
Transformer (big)	**28.4**	**41.8**	$2.3 \cdot 10^{19}$	

BERT (Devlin et al. 2019)



https://arxiv.org/abs/1810.04805

Why is BERT worth knowing? (150K+ citations)

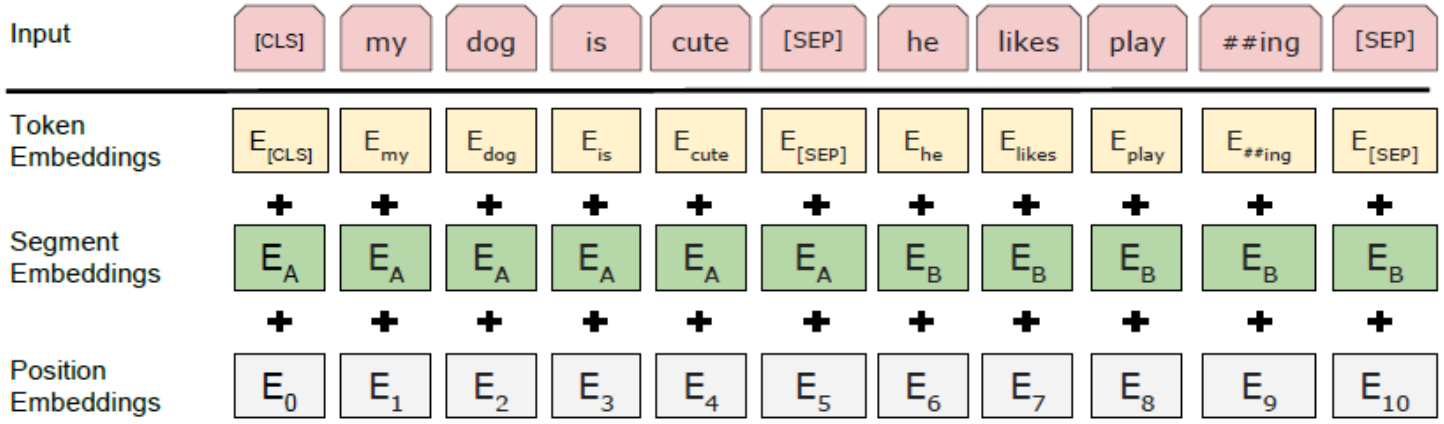
BERT is a significant breakthrough in NLP

- One of the first LLMs
- Transformer-based architecture
- Pre-training on large amounts of text data via masking objective
- Fine-tuned to achieve SotA for many tasks
- Still widely used

Overview of BERT

- Uses standard transformer blocks, encoder only
  - Base: 12 layers, 768-dim, 12 heads; 110M parameters
  - Large: 24 layers, 1024-dim, 16 heads; 340M parameters
- WordPiece: 30K tokens
  - Special [CLS] and [SEP] tokens
  - Positional and sentence embeddings
- Pre-trained with masked language modeling (MLM) and next sentence prediction
- Fine-tuned for other tasks

BERT: Input representation



- WordPiece: 30K tokens
  - Special tokens
    - [CLS] at start to encode sentence summary
    - [SEP] to separate sentences or question and answer
  - Positional and sentence embeddings

“Encoder-only” and “Decoder-only” architectures

Encoder-only (e.g. BERT): Each token attends to all input tokens (**bidirectional**). May output a single vector (CLS) or one per token.

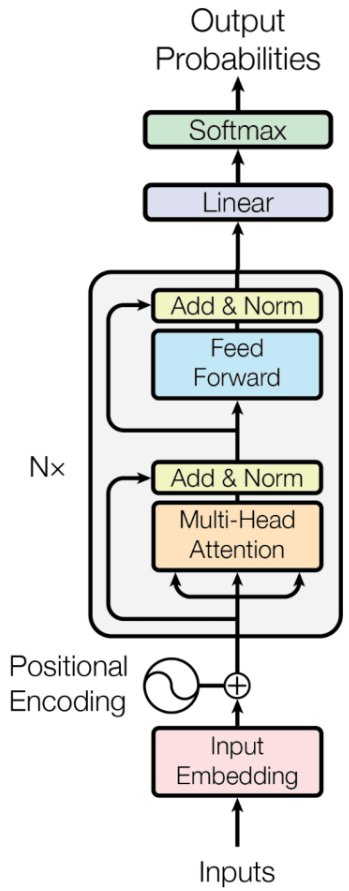


Figure: https://towardsdatascience.com/a-complete-guide-to-bert-with-code-9f87602e4a11/

Decoder-only (e.g. GPT): Each token attends to only the preceding tokens (**causal**) and outputs a prediction of the next token.

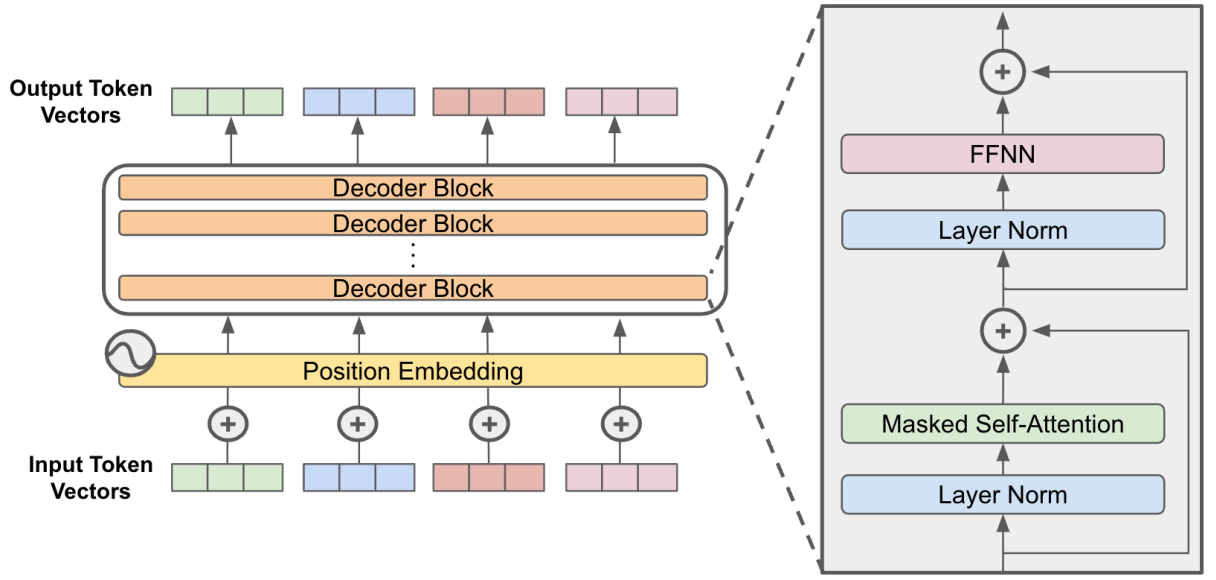
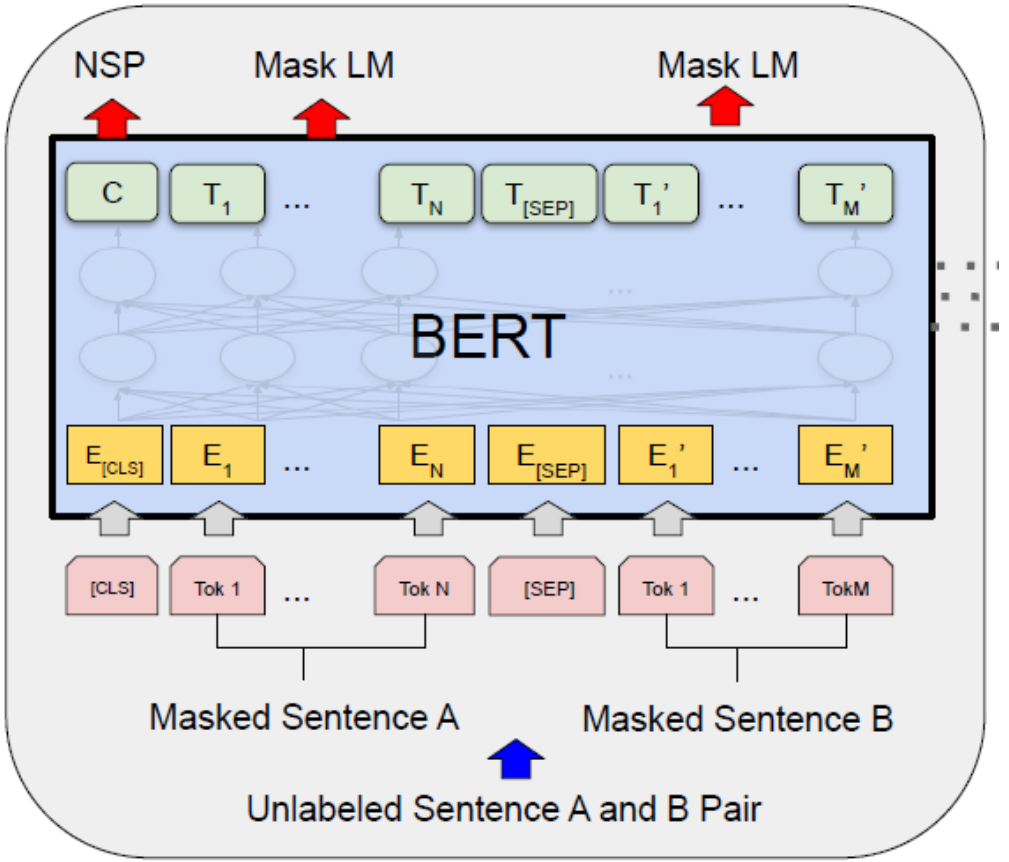


Figure: https://cameronrwolfe.substack.com/p/decoder-only-transformers-the-workhorse

Pre-training with masked language modeling (MLM)

- Randomly select 15% of text tokens to be “masked” and predicted by based on surrounding tokens
- Masked tokens are replaced by
  - [MASK] token (80% of the time)
  - Random token (10%)
  - Unchanged token (10%)
- Only masked tokens are evaluated



Pre-training

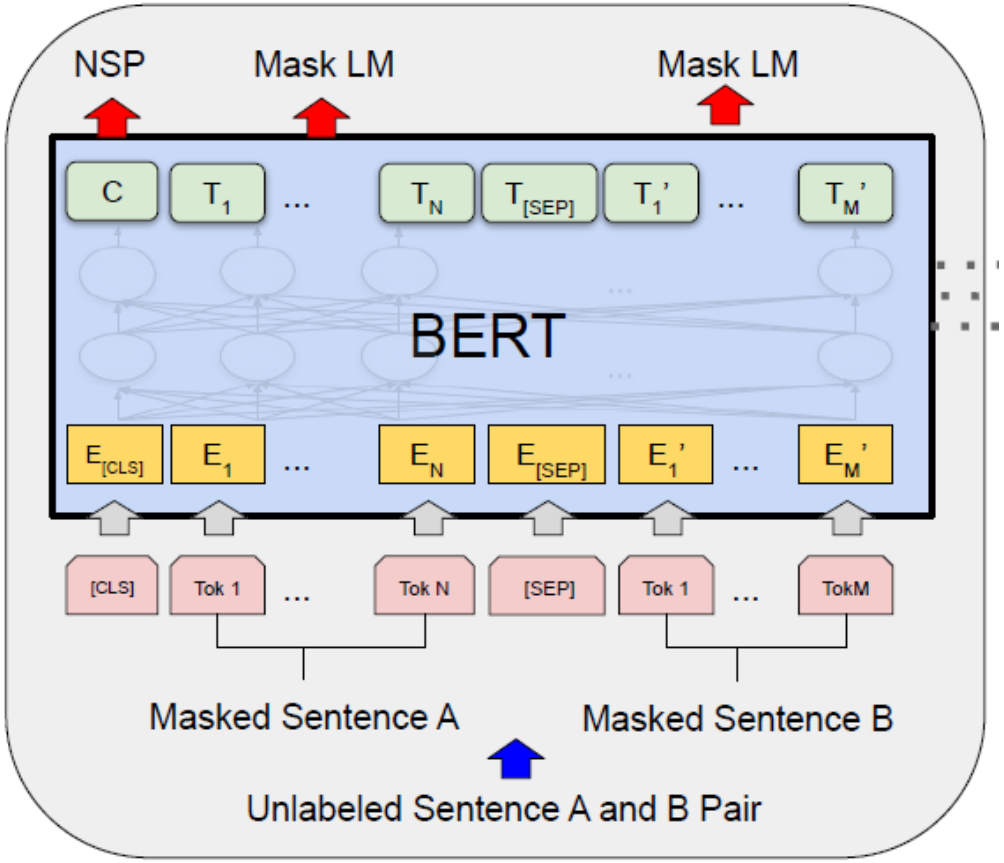
Masking example:

[cls] ***My*** dog is ***cute*** [sep] He ***likes*** play ##ing [sep]

[cls] **[MASK]** dog is **grave** [sep] He ***likes*** play ##ing [sep]

Pre-training with next sentence prediction

- Input is two sentences A and B
- Replace B with a random sentence 50% of the time
- Predict whether B is the original sentence or not (via [CLS] token)



Pre-training

Pre-training and fine-tuning

- Pre-train on MLM and NSP tasks
  - BooksCorpus: 800M words
  - English Wikipedia: 2.5B words
  - Important to use full documents, not just shuffled sentences
  - $\text{BERT}_{\text{BASE}}$ trained on 16 TPU chips; $\text{BERT}_{\text{LARGE}}$ on 64 chips; 4 days each
- Fine-tune on each task, takes a few hours on a GPU
  - Paraphrasing
  - Entailment
  - Question answering
  - …

BERT results

GLUE: General Language Understanding Evaluation – many tasks

System	MNLI-(m/mm) 392k	QQP 363k	QNLI 108k	SST-2 67k	CoLA 8.5k	STS-B 5.7k	MRPC 3.5k	RTE 2.5k	Average -
Pre-OpenAI SOTA	80.6/80.1	66.1	82.3	93.2	35.0	81.0	86.0	61.7	74.0
BiLSTM+ELMo+Attn	76.4/76.1	64.8	79.8	90.4	36.0	73.3	84.9	56.8	71.0
OpenAI GPT	82.1/81.4	70.3	87.4	91.3	45.4	80.0	82.3	56.0	75.1
$\text{BERT}_{\text{BASE}}$	84.6/83.4	71.2	90.5	93.5	52.1	85.8	88.9	66.4	79.6
$\text{BERT}_{\text{LARGE}}$	**86.7/85.9**	**72.1**	**92.7**	**94.9**	**60.5**	**86.5**	**89.3**	**70.1**	**82.1**

Table 1: GLUE Test results, scored by the evaluation server (https://gluebenchmark.com/leaderboard). The number below each task denotes the number of training examples. The "Average" column is slightly different than the official GLUE score, since we exclude the problematic WNLI set.[8] BERT and OpenAI GPT are single-model, single task. F1 scores are reported for QQP and MRPC, Spearman correlations are reported for STS-B, and accuracy scores are reported for the other tasks. We exclude entries that use BERT as one of their components.

MNLI: whether a sentence entails, contradicts, or is unrelated to another
QQP: whether two sentences are semantically equivalent
QNLI: question answering
SST-2: positive or negative sentiment
CoLA: whether sentence is grammatically correct
STS-B: sentence similarity score
MRPC: whether one sentence paraphrases another
RTE: whether a sentence entails a hypothesis

SQuAD: question answering dataset

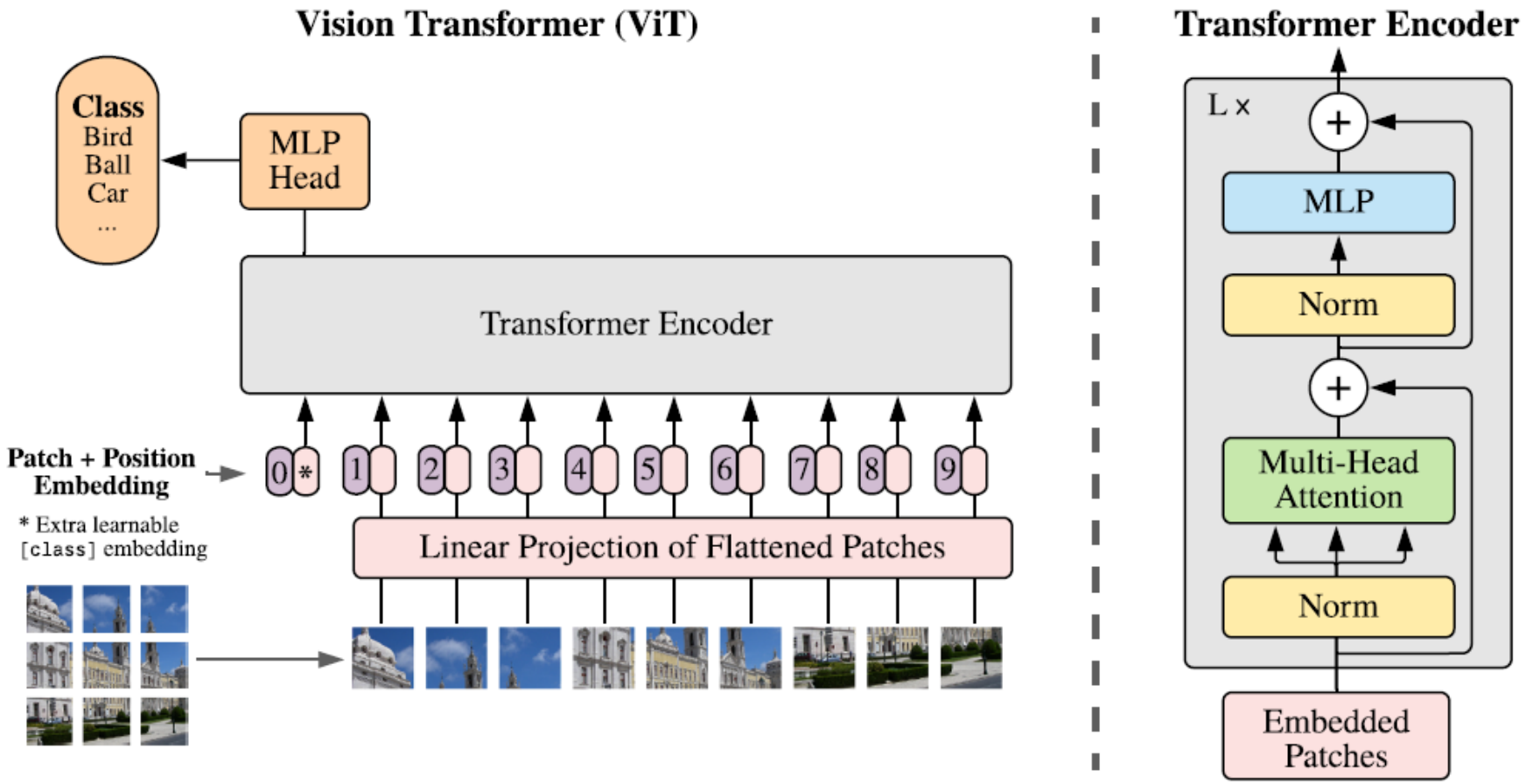
System	Dev EM	Dev F1	Test EM	Test F1
Top Leaderboard Systems (Dec 10th, 2018)				
Human	-	-	82.3	91.2
#1 Ensemble - nlnet	-	-	86.0	91.7
#2 Ensemble - QANet	-	-	84.5	90.5
Published				
BiDAF+ELMo (Single)	-	85.6	-	85.8
R.M. Reader (Ensemble)	81.2	87.9	82.3	88.5
Ours				
$\text{BERT}_{\text{BASE}}$ (Single)	80.8	88.5	-	-
$\text{BERT}_{\text{LARGE}}$ (Single)	84.1	90.9	-	-
$\text{BERT}_{\text{LARGE}}$ (Ensemble)	85.8	91.8	-	-
$\text{BERT}_{\text{LARGE}}$ (Sgl.+TriviaQA)	**84.2**	**91.1**	**85.1**	**91.8**
$\text{BERT}_{\text{LARGE}}$ (Ens.+TriviaQA)	**86.2**	**92.2**	**87.4**	**93.2**

Table 2: SQuAD 1.1 results. The BERT ensemble is 7x systems which use different pre-training checkpoints and fine-tuning seeds.

Key Take-aways from BERT

- Bi-directional masked language modeling is highly effective pre-training
  - Does not require supervision
  - Learns general representations
- Same idea has been adopted for vision, but with much higher masking ratio (~80% of patches masked)

ViT: Vision Transformers (Dosovitskiy et al. 2021)



https://arxiv.org/abs/2010.11929

Why is ViT worth knowing about? (100K+ citations)

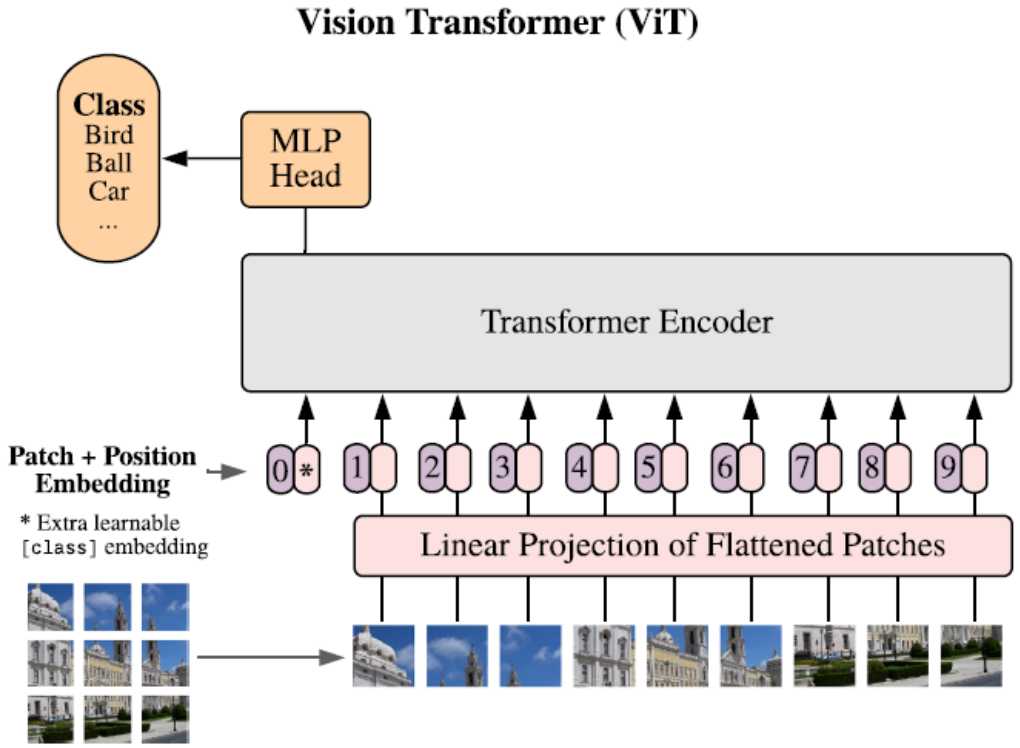
1. Shows that the same exact Transformer blocks can be used for vision, paving the way for multimodal processing

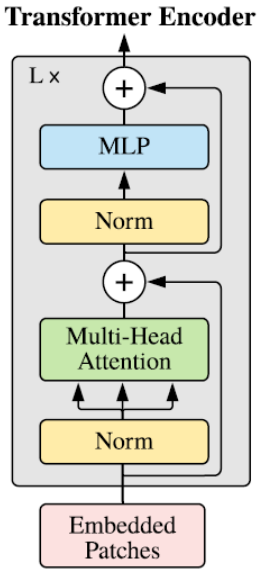
2. Transformers work as well as CNNs but are more computationally efficient

3. Vision and language are easier to combine using transformers

ViT Overview

- Image is divided into patches (e.g., 16x16)
- Each patch projects into a fixed length vector
- Positional encoding added to each patch
- Extra [CLS] token to encode image summary
- Multiple layers of standard transformer (same as for language)
- For classification, final prediction is linear layer applied to [class] token





Different size models, same pretraining

Same model, different pretraining dataset

Big convolutional networks

	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** ± 0.04	87.76 ± 0.03	85.30 ± 0.02	87.54 ± 0.02	88.4/88.5*
ImageNet ReaL	**90.72** ± 0.05	90.54 ± 0.03	88.62 ± 0.05	90.54	90.55
CIFAR-10	**99.50** ± 0.06	99.42 ± 0.03	99.15 ± 0.03	99.37 ± 0.06	–
CIFAR-100	**94.55** ± 0.04	93.90 ± 0.05	93.25 ± 0.05	93.51 ± 0.08	–
Oxford-IIIT Pets	**97.56** ± 0.03	97.32 ± 0.11	94.67 ± 0.15	96.62 ± 0.23	–
Oxford Flowers-102	99.68 ± 0.02	**99.74** ± 0.00	99.61 ± 0.02	99.63 ± 0.03	–
VTAB (19 tasks)	**77.63** ± 0.23	76.28 ± 0.46	72.72 ± 0.21	76.29 ± 1.70	–
TPUv3-core-days	2.5k	0.68k	0.23k	9.9k	12.3k

$30K

$120K

Model	Layers	Hidden size D	MLP size	Heads	Params
ViT-Base	12	768	3072	12	86M
ViT-Large	24	1024	4096	16	307M
ViT-Huge	32	1280	5120	16	632M

Table 1: Details of Vision Transformer model variants.

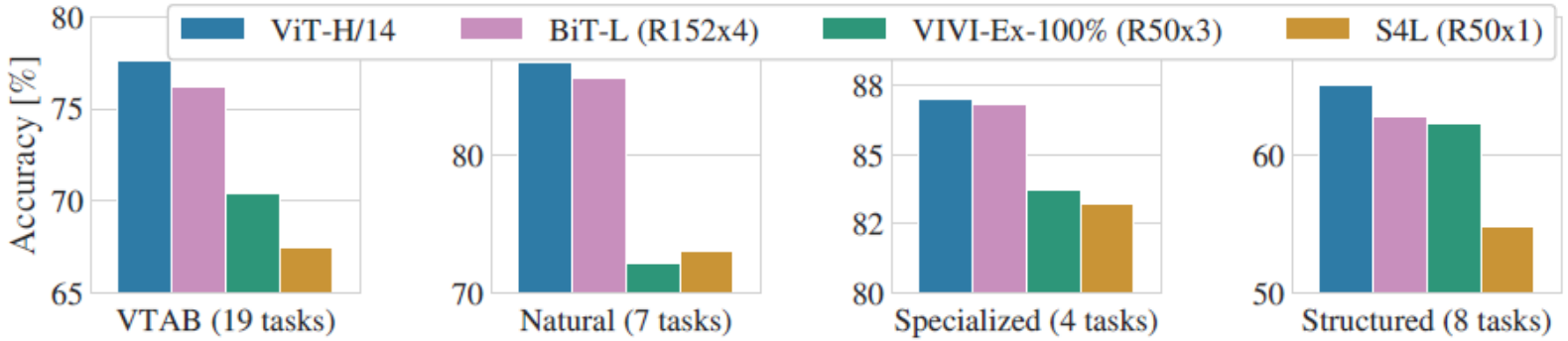
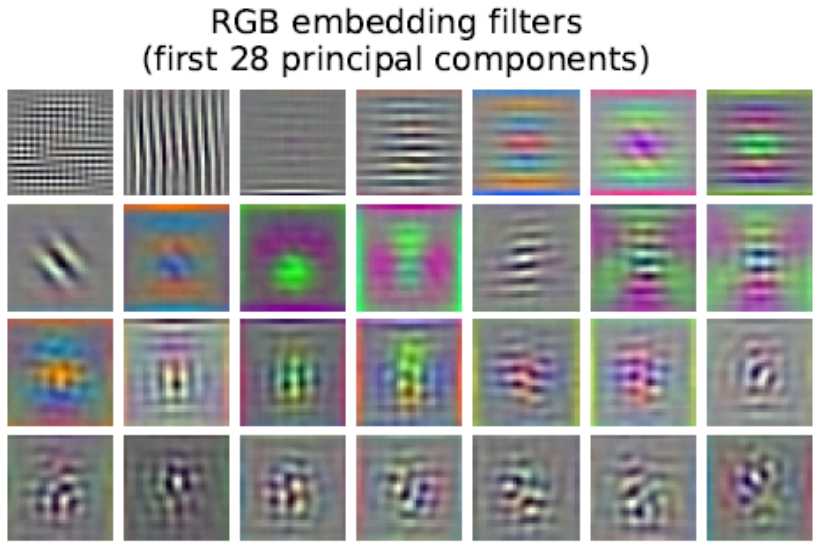
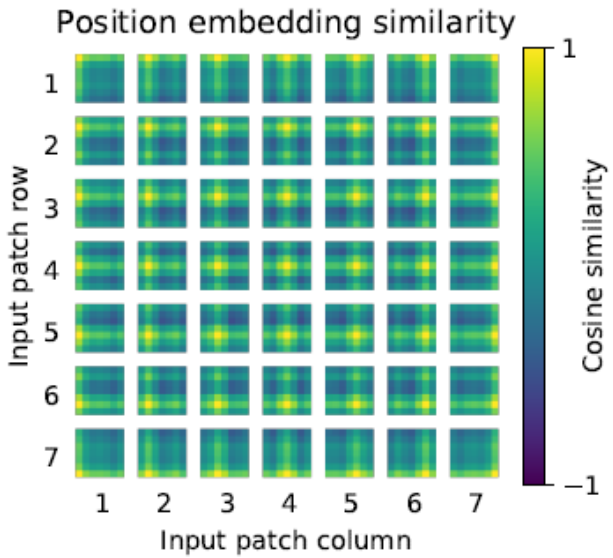


Figure 2: Breakdown of VTAB performance in *Natural*, *Specialized*, and *Structured* task groups.

Information is integrated among distant patches





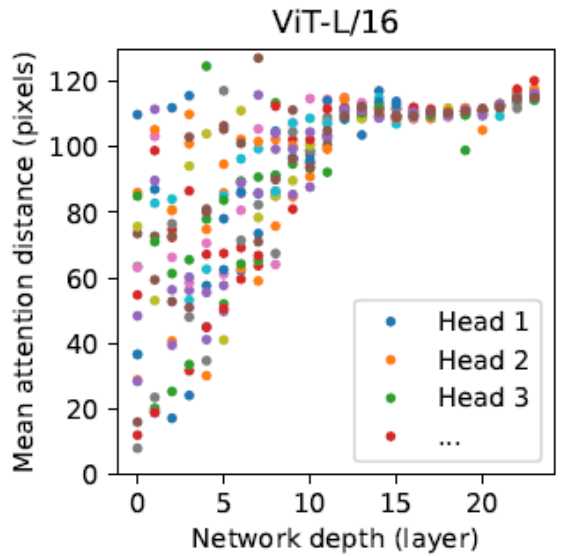
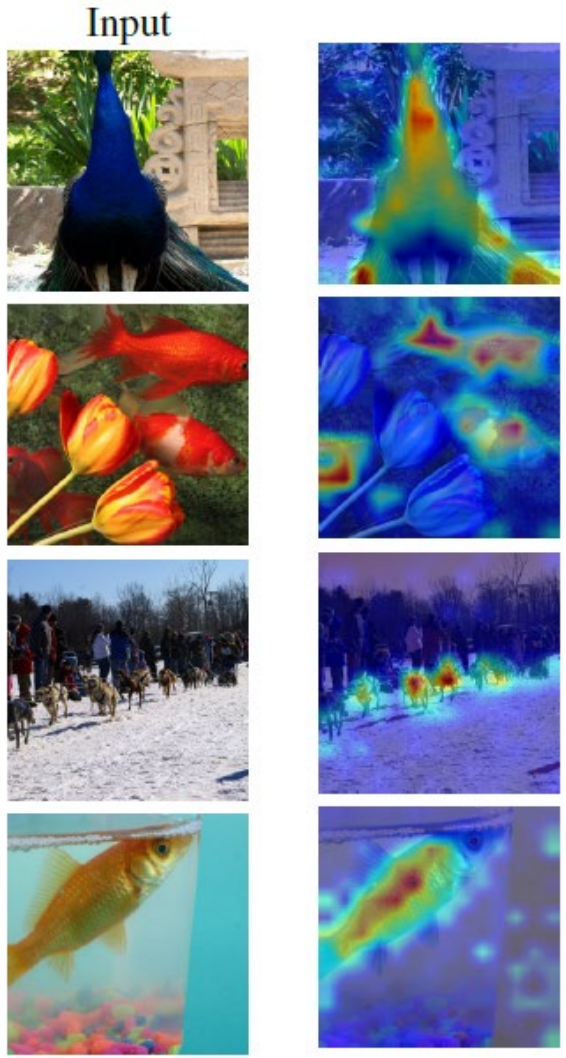
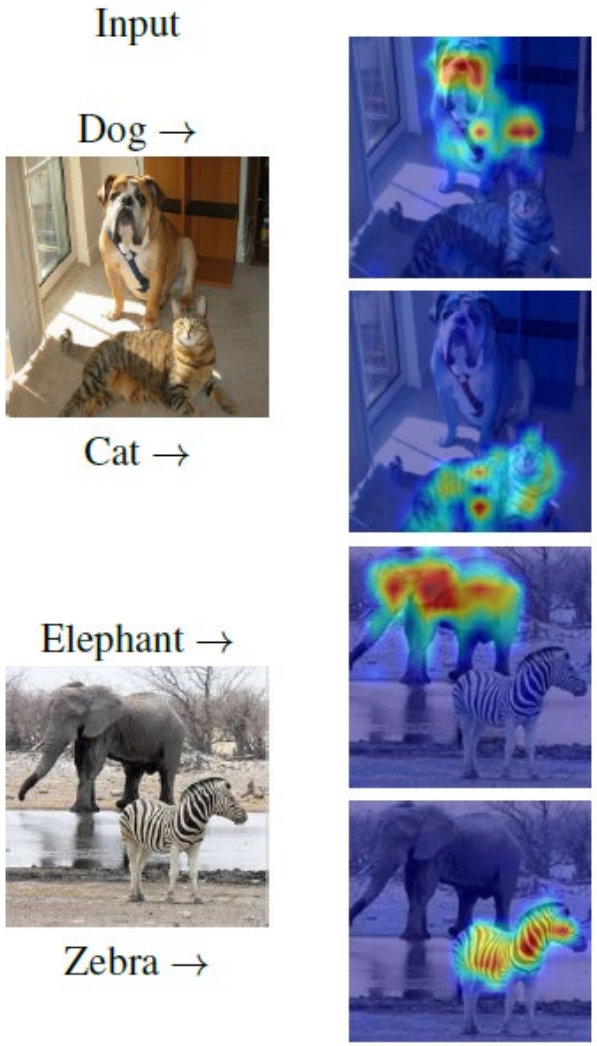


Figure 7: **Left:** Filters of the initial linear embedding of RGB values of ViT-L/32. **Center:** Similarity of position embeddings of ViT-L/32. Tiles show the cosine similarity between the position embedding of the patch with the indicated row and column and the position embeddings of all other patches. **Right:** Size of attended area by head and network depth. Each dot shows the mean attention distance across images for one of 16 heads at one layer. See Appendix D.7 for details.

Visualizing transformer (Chefer et al. 2021)

Visualizations for overall and class-specific feature importance





CNNs vs. Transformers

- **CNNs** encode position as an index in the feature map. **Transformers** do not care about index order but encode positional embeddings
  - Even when positional embeddings are not used, transformer models still surprisingly work well (at least for vision)

- **CNNs** encode a bias that nearby pixels are most related. **Transformers** enable combining information from distant patches, with positional embedding making it easy to learn that nearby patches are related
  - CNNs can only use information in neighboring pixels/cells, but the receptive field (pixel area considered) grows larger as network gets deeper

- In practice, CNNs and Transformers perform similarly for pure vision tasks given enough data; CNNs do better with less data, while Transformers scale more efficiently with large data and compute
  - Hybrids are possible, e.g. apply shallow CNN before first patch embedding

- Transformers operate on “tokens”, which is very general and can be applied to any modality

Q2-4

https://tinyurl.com/AML441-L19

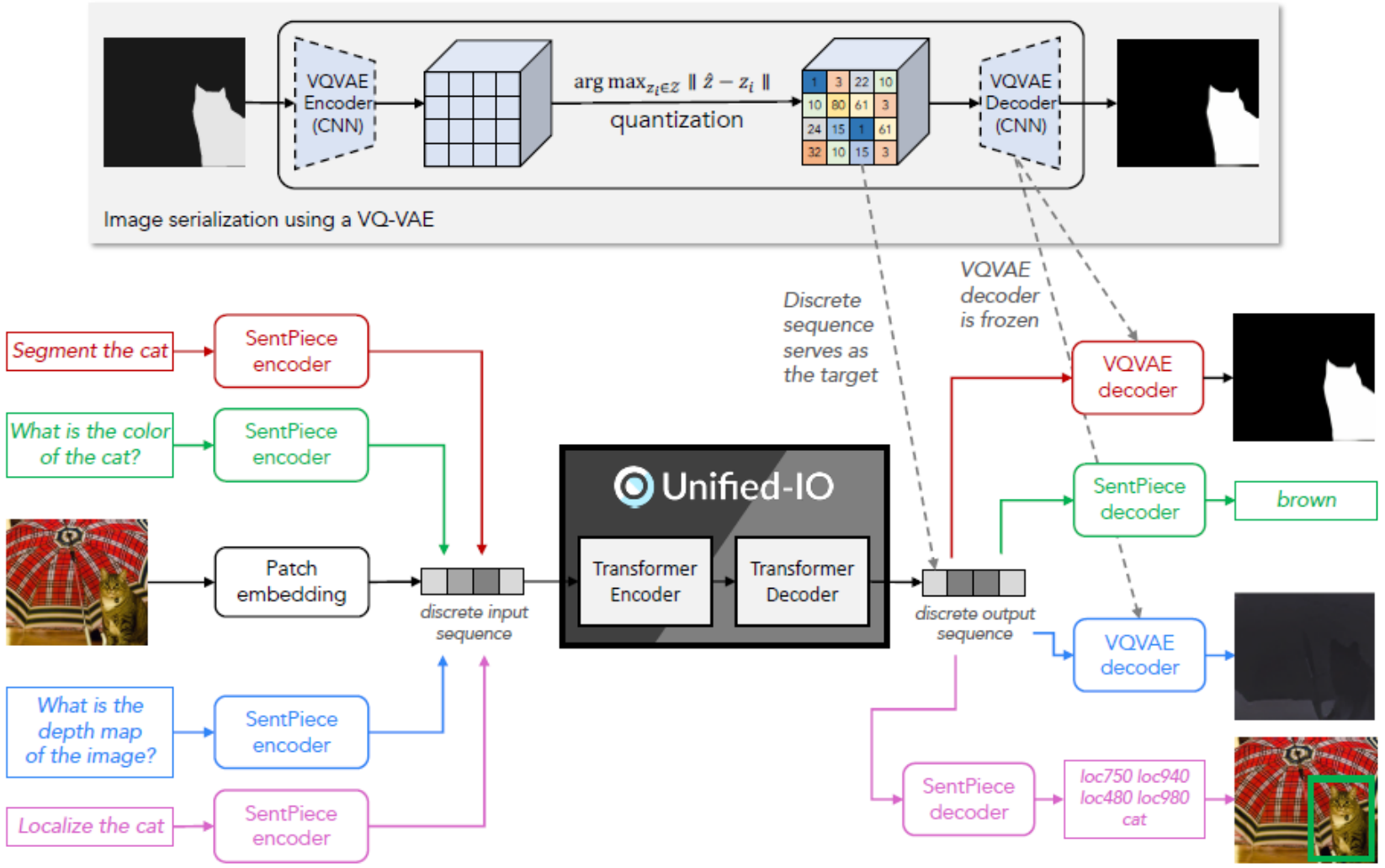


Unified-IO: <text, image> to <text, image> (Lu et al. 2022)

3B parameters

Pre-train on masked text and image completion for text, images, and image/caption pairs

Multitask training on 80 datasets



Unified-IO (June 2022)

https://unified-io.allenai.org/

Vision tasks

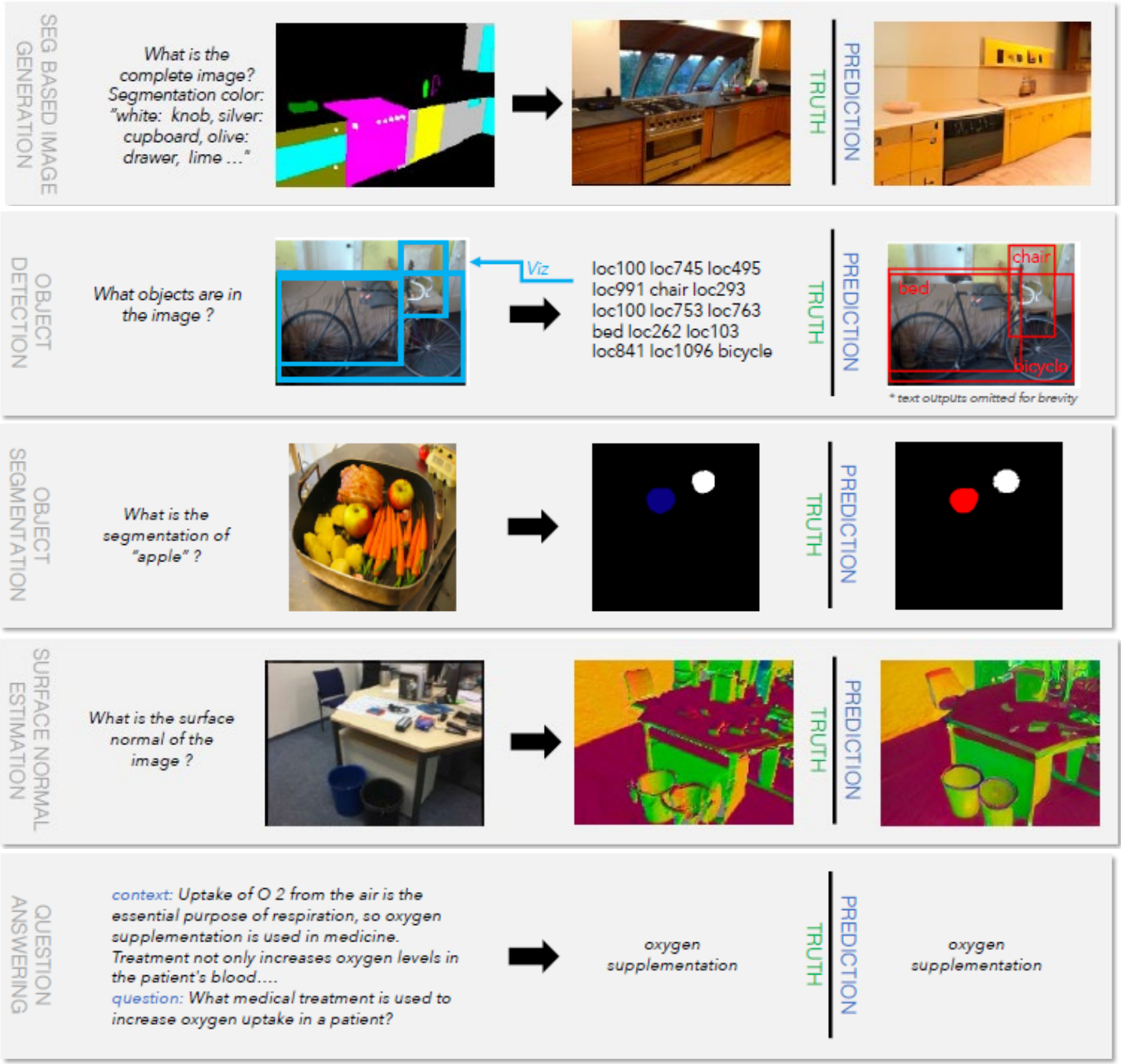
- Image synthesis from text / inpainting / segmentation
- Image/object classification
- Object detection, segmentation, keypoint estimation
- Depth/normal estimation

Vision-language tasks

- VQA, image/region captioning, referring expressions comprehension, relationship detection

NLP tasks

- Question answering
- Text classification



Often performs similarly or better than SotA single-task models

	NYUv2	*ImageNet*	*Place365*	*VQAv2*	*OkVQA*	*A-OkVQA*	*VizWizQA*	*VizWizGround*	*Swig*	*SNLI-VE*	*VisComet*	*Nocaps*	*COCO*	*COCO*	*MRPC*	*BoolQ*	*SciTail*
Split	val	val	val	test-dev	test	test	test-dev	test-std	test	val	val	val	val	test	val	val	test
Metric	RMSE	Acc.	Acc.	Acc.	Acc.	Acc.	Acc.	IOU	Acc.	Acc.	CIDEr	CIDEr	CIDEr	CIDEr	F1	Acc	Acc
Unified SOTA	UViM	-	-	-	Flamingo	-	Flamingo	-	-	-	-	-	-	-	T5	PaLM	-
	0.467	-	-	-	57.8	-	49.8	-	-	-	-	-	-	-	92.20	92.2	-
UNIFIED-IO$_{\text{SMALL}}$	0.649	42.8	38.2	57.7	31.0	24.3	42.4	35.5	17.3	76.5	-	45.1	80.1	-	84.9	65.9	87.4
UNIFIED-IO$_{\text{BASE}}$	0.469	63.3	43.2	61.8	37.8	28.5	45.8	50.0	29.7	85.6	-	66.9	104.0	-	87.9	70.8	90.8
UNIFIED-IO$_{\text{LARGE}}$	0.402	71.8	50.5	67.8	42.7	33.4	47.7	54.7	40.4	86.1	-	87.2	117.5	-	87.5	73.1	93.1
UNIFIED-IO$_{\text{XL}}$	0.385	79.1	53.2	77.9	54.0	45.2	57.4	65.0	49.8	91.1	21.2	100.0	126.8	122.3	89.2	79.7	95.7
Single or fine-tuned SOTA	BinsFormer	CoCa	MAE	CoCa	KAT	GPV2	Flamingo	MAC-Caps	JSL	OFA	SVT	CoCa	-	OFA	Turing NLR	ST-MOE	DeBERTa
	0.330	91.00	60.3	82.3	54.4	38.1	65.7	27.3	39.6	91.0	18.3	122.4	-	145.3	93.8	92.4	97.7

Explore demo of Unified IO 2 which extends to audio and multiple image inputs

https://unified-io-2.allenai.org/

HW 5 (Coming soon)

1. Applications of AI [30 pts]

Choose an application area (e.g. AI for agriculture, advertising, recommendation systems, self-driving cars/trucks, manufacturing, construction, home monitoring, virtual assistants), read at least two related media articles or papers, cite your sources, and answer the following questions:

1. How is AI used in that application area? What is the problem that AI is trying to solve, and what are the key AI/ML technologies involved? What are the technical challenges? (100+ words)
2. What is the actual or potential positive impact? Who is impacted? (50+ words)
3. What is the actual or potential negative impact? Who is impacted? (50+ words)

If you use media articles, they should be from highly respected sources, e.g. The Economist, Wall Street Journal, NY Times, Washington Post, Atlantic, MIT Technology Review. This will be graded as complete/incomplete. Any citation format is OK, as long as it clearly indicates the source, article title, and date of article.

2. Fine-tune ImageNet Model for Pets Classification [30 pts]

A convolutional neural network (CNN) is a type of deep learning artificial neural network commonly used for image processing tasks. In this section, you will load a pretrained network, finetune it for a new task, and evaluate it to achieve a good image classification accuracy on the **Oxford-IIIT Pet dataset**. The task is to classify the breed of cat or dog.

This code is worth understanding, but writing it all from scratch would be too much work for those new to deep network learning. So we have provided most of the code structure. Read it carefully and uncomment it as you go, where indicated. Your key steps are:

1. Load a pretrained ResNet34 and replace the last layer of the network so that the output dimension matches the number of Pet classes. See Mar 26 lecture. Include the printout of network structure and parameters in your report.
2. Set the learning rate and learning scheduler, train, and evaluate. You may need to modify the training settings to get a better performance. You can consider the data augmentation, learning rate, learning rate scheduler, and batch size. Attach the accuracy and loss plots for training_set and test_set, and report your best testing accuracy. You should be able to reach an accuracy of at least 90% within 10 epochs.

3. Zero-shot and Linear Probe using CLIP [40 pts]

CLIP (Contrastive Language-Image Pre-Training), developed by OpenAI, is a neural network trained on 400 million (image, text) pairs. CLIP encodes text and images into the same vector space, so that it can match images to text descriptions by encoding each and computing the dot product of the encodings. This enables zero-shot classification, i.e. assigning images to text labels from a dataset without training on that dataset. The following are some resources you may find useful for understanding CLIP and finishing your tasks in this homework.

- GitHub Repo (Make sure to look at this for examples of how to use CLIP)
- Learning Transferable Visual Models From Natural Language Supervision (paper)
- CLIP Website
- Labels for Flowers 102 dataset (json file)

The starter code includes sections for loading the Flowers 102 dataset, the CLIP model, and examples of usage. The code splits the data into a "train" and "test" (validation) set.

Your tasks are to use CLIP for a new classification task in three ways:

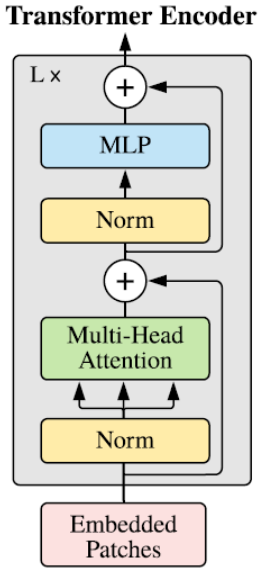
Zero-shot: Use pre-trained text and image representations to classify images. For zero-shot recognition (no training), text features are computed from the CLIP model for phrases such as "An image of <flower_name>, a type of flower" for varying <flower_name> inserts. Then, image features are computed using the CLIP model for an image, and the cosine similarity between each text and image is computed. The label corresponding to the most similar text is assigned to the image. We've included an example in the starter code. You'll get that working using a data loader, which enables faster batch processing; then, compute the accuracy over the test set. You should see top-1 accuracy (i.e. percent of times the highest confidence predicted label is correct) in the 60-70% range.

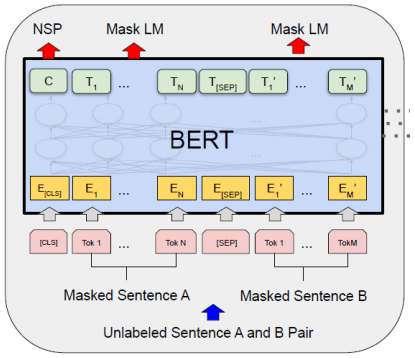
Linear probe: Often, it is possible to improve performance for a particular classification task by training a linear classifier on the model's features. This is called a "linear probe". We can do that by extracting the CLIP pre-trained image features and then training a linear logistic regression classifier. We do not use text features for the linear probe method. Train on the train set, and evaluate on the test set and report your performance. You can get top-1 accuracy in the 90-95% range. If you see accuracy in the 80's, try both normalizing and not normalizing the features.

Nearest neighbor: We can also extract pretrained CLIP features and apply a nearest neighbor classifier. You can use your own implementation of nearest neighbor or a library like sklearn or FAISS for this. Try K={1, 3, 5, 7, 11, 21}. Report performance for best K on the test set. You should see top-1 accuracy in the 80-90% range.

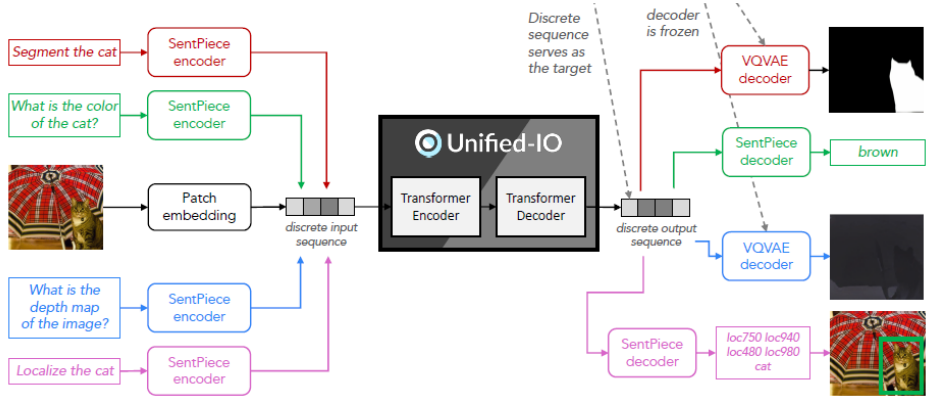
Things to remember

- Transformers are general data processors, applicable to text, vision, audio, control, and other domains
- Pre-training to generate missing tokens in unsupervised text data learns a general model that can be fine-tuned
  - Same idea is also applicable to other domains
- Transformer architectures are state-of-art for vision and language individually
- Arguably, the biggest benefit of transformers is ability to combine information from multiple domains





Pre-training



Next class: Foundation Models

- CLIP
- GPT line of work