Generative Pretraining from Pixels

Mark Chen, Alec Radford, Rewon Child, Jeff Wu, Heewoo Jun, Prafulla Dhariwal,
David Luan, Ilya Sutskever

ICML 2020

Presented by Mingyu Yang

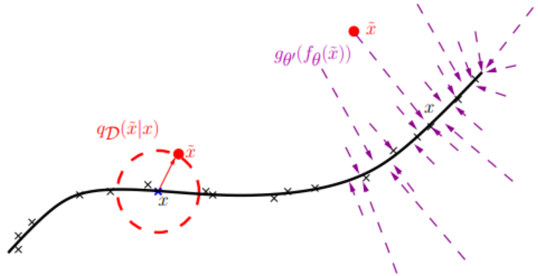
Background

- **Unsupervised _generative_ pre-training for images:**
  - Popular in *mid 2000's*
  - A central role in the *resurgence of deep learning*:
    - Before that, DNNs are very hard to train!
    - People believe that learning $p(x)$ helps supervised modeling of $p(y|x)$
    - Pre-training + fine-tuning achieves STOA performance and outperforms SVM in MNIST

Background

- **Unsupervised *generative* pre-training for images:**
  - Popular in *mid 2000's*
  - A central role in the *resurgence of deep learning*:
    - Before that, DNNs are very hard to train!
    - People believe that learning $p(x)$ helps supervised modeling of $p(y|x)$
    - Pre-training + fine-tuning achieves STOA performance and outperforms SVM in MNIST
  - Example:
    - Deep Belief Network (2006)
    - Denoising Autoencoder (2008)

Digits generated from Deep Belief Networks



Manifold learning perspective of denoising autoencoder

Background

- **Unsupervised *generative* pre-training becomes less popular for images:**

    1. Deep Neural Networks are *much easier to train*
        - Better activation functions: *ReLU, LeakyReLU, …*
        - Improved initializations: *Xavier, Kaiming, …*
        - Normalization strategies: *Batch Normalization …*

Background

- **Unsupervised *generative* pre-training becomes less popular for images:**

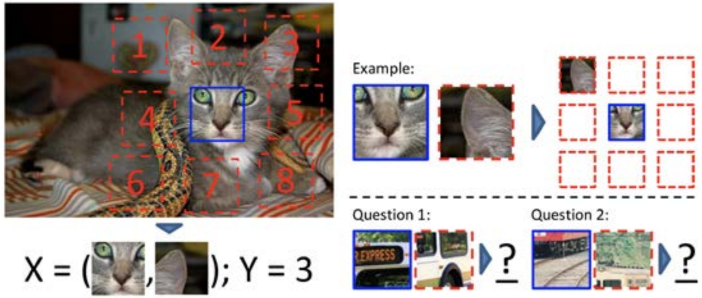
1. Deep Neural Networks are *much easier to train*
   - Better activation functions: *ReLU, LeakyReLU, …*
   - Improved initializations: *Xavier, Kaiming, …*
   - Normalization strategies: *Batch Normalization …*

2. *Supervised pre-training* achieves better performance

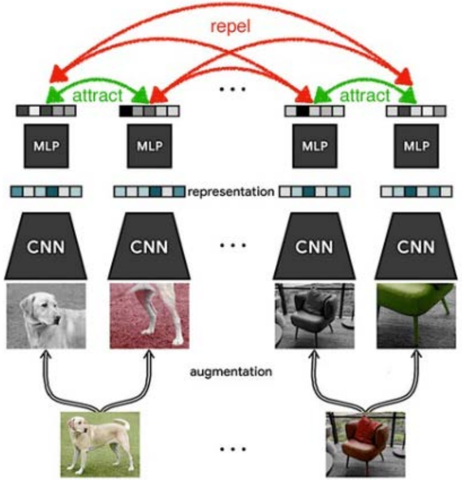
Background

- **Unsupervised *generative* pre-training becomes less popular for images:**

3. People design different *self-supervised learning* methods model *global structures* (by solving pretext tasks) instead of the distribution
   - Predict relative positions
   - Mutual Information: *AMDIM*
   - Contrastive learning: *MoCo, SimCLR*



Predicting relative positions (Doersch et al., 2015)



SimCLR (Chen et al., 2019)

Background

- **Unsupervised generative pre-training *<u>flourished in NLP!</u>***
  - Learn the language model as pre-training
    - BERT (2018): $L_{BERT} = \mathop{\mathbb{E}}_{x \sim X} \mathop{\mathbb{E}}_{M} \sum_{i \in M} \left[-\log p \left(x_i | x_{[1,n] \setminus M} \right) \right]$
    - GPT-2 (2019), GPT-3 (2020): $L_{AR} = \mathop{\mathbb{E}}_{x \sim X} [-\log p(x)]$

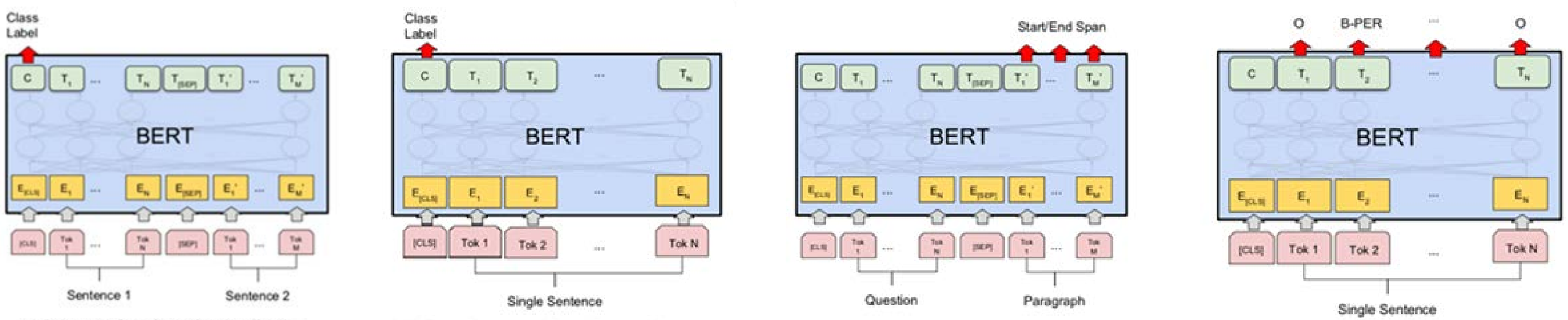
Predict masked words: **Conditional probability**

$$p(x) = \prod_{i=1}^{n} p(x_{\pi_i} | x_{\pi_1}, ..., x_{\pi_{i-1}}, \theta)$$

Autoregressive model

Background

- **Unsupervised generative pre-training *flourished in NLP!***
  - Fine-tuning for different tasks



(a) Sentence Pair Classification Tasks: MNLI, QQP, QNLI, STS-B, MRPC, RTE, SWAG

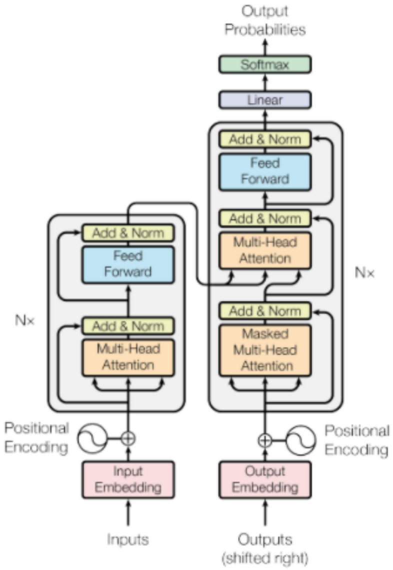
(b) Single Sentence Classification Tasks: SST-2, CoLA

(c) Question Answering Tasks: SQuAD v1.1

(d) Single Sentence Tagging Tasks: CoNLL-2003 NER

Background

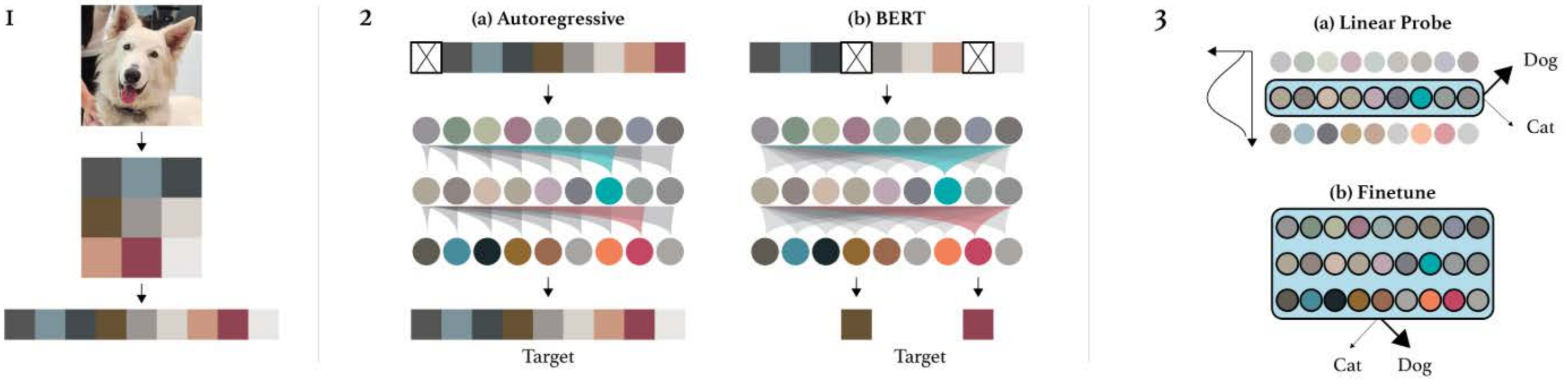
- **Unsupervised generative pre-training *flourished in NLP!***
  - STOA methods benefit from *attention mechanism*
    - Transformer (2017)
    - Will be introduced later in this course
  - Motivation of this paper:
    - Can we do the same to images?
    - Can we get competitive performance?
  - Very similar to BigBiGAN presented last class
    - Autoregressive vs GAN



Transformer architecture

Method

- **Overview**



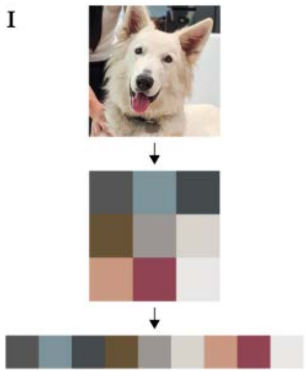
Pre-processing

Pre-training

Fine-tuning

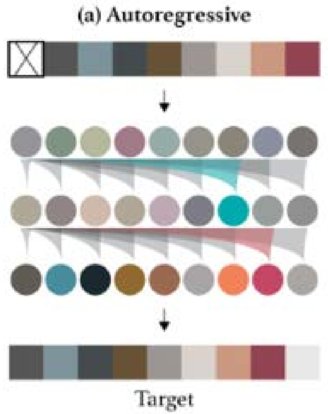
Method

- **Pre-processing:**
  - Context Reduction:
    - Images are *<u>so large for transformers</u>*
      - ImageNet: 224x224x3
    - Downsampling:
      - Reduce the size to 32x32x3, 48x48x3, or 64x64x3
    - Reduce 3 dimensional (R,G,B) channels to 1 dimensional using K-means
      - 512 clusters
      - Further reduce the size to 32x32, 48x48, or 64x64



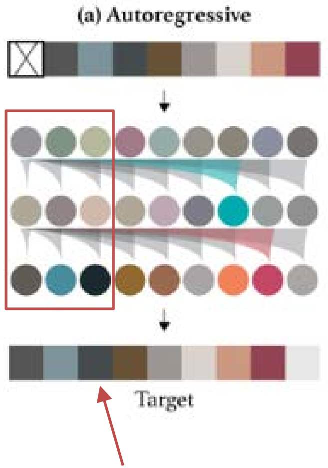
Method

- **Pre-training:**
  - Autoregressive (AR):
    - Likelihood: $p(x) = \prod_{i=1}^{n} p(x_{\pi_i} | x_{\pi_1}, ..., x_{\pi_{i-1}}, \theta)$
    - Minimize log-likelihood: $L_{AR} = \mathbb{E}_{x \sim X} [-\log p(x)]$
    - Raster order
    - Upper triangular mask to zero out the effect of future words (pixels)



Method

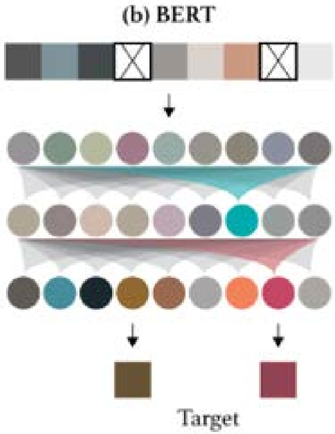
- **Pre-training:**
  - Autoregressive (AR):
    - Likelihood: $p(x) = \prod_{i=1}^{n} p(x_{\pi_i} | x_{\pi_1}, ..., x_{\pi_{i-1}}, \theta)$
    - Minimize log-likelihood: $L_{AR} = \mathbb{E}_{x \sim X}[-\log p(x)]$
    - Raster order
    - Upper triangular mask to zero out the effect of future words (pixels)



Method

- **Pre-training:**
  - BERT:
    - Minimize the negative log-likelihood of the masked elements conditioned on the unmasked ones

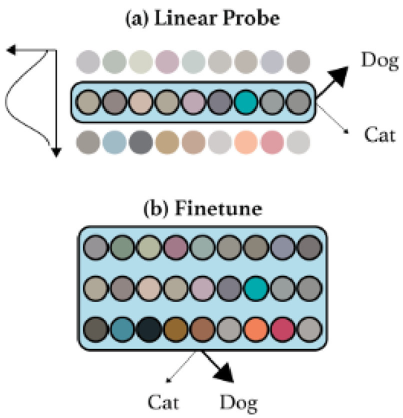
$$L_{BERT} = \mathop{\mathbb{E}}_{x \sim X} \mathop{\mathbb{E}}_{M} \sum_{i \in M} \left[-\log p\left(x_i | x_{[1,n] \setminus M}\right) \right]$$



Method

- **Fine-tuning:**
  - At each layer, each pixel gets a ***d-dimensional*** feature vector
  - ***Average pooling*** across the sequence dimension to get a ***d-dimensional*** feature vector for the whole image
  - Linear Probe & Fine-tuning

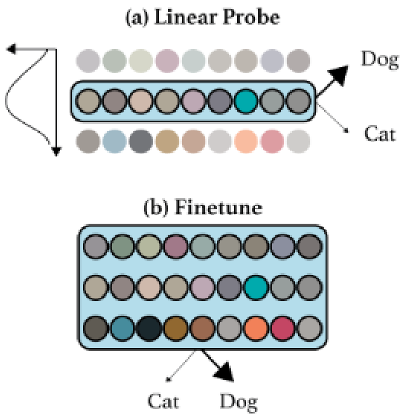
3



Method

- **Fine-tuning:**
  - Linear Probe (transfer learning):
    - Treat the transformer as a *fixed feature extractor*
    - Learn a projection to class logits and minimize the cross entropy loss L_{CLF}
    - Could extract the features at any intermediate layer

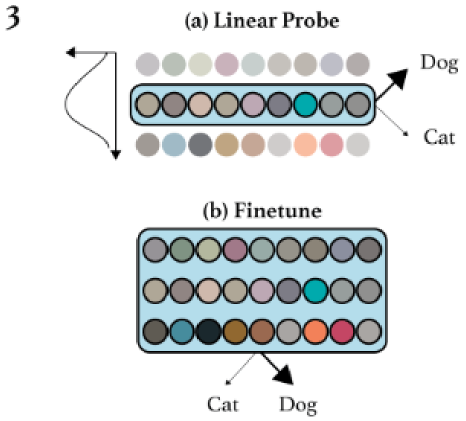
3



Method

- **Fine-tuning:**

  – Fine-tuning:
    - Treat the learned transformer as *an initialization*
    - Learn a projection to class logits and minimize the joint objective $L_{GEN} + L_{CLF}$



Method

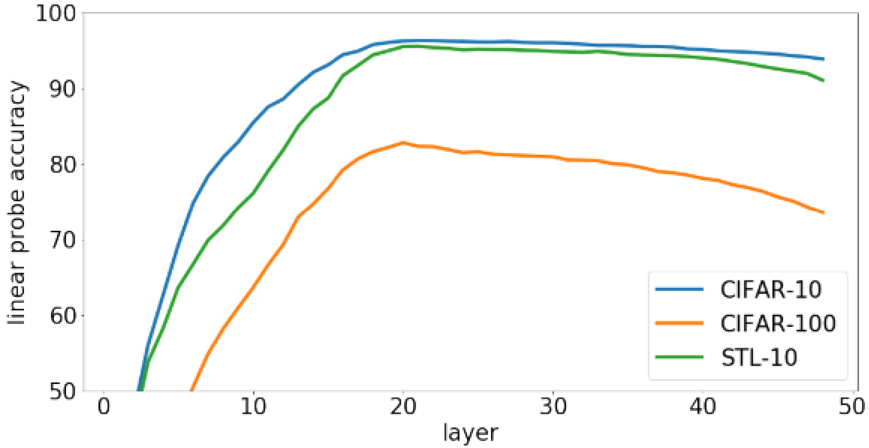
- **More details:**
  - Same model with GPT-2 with slight modification

	# of layers	Embedding size	# of parameters
iGPT-S	24	512	76M
iGPT-M	36	1024	455M
iGPT-L	48	1536	1.4B
iGPT-XL	60	3072	6.8B

Model parameters

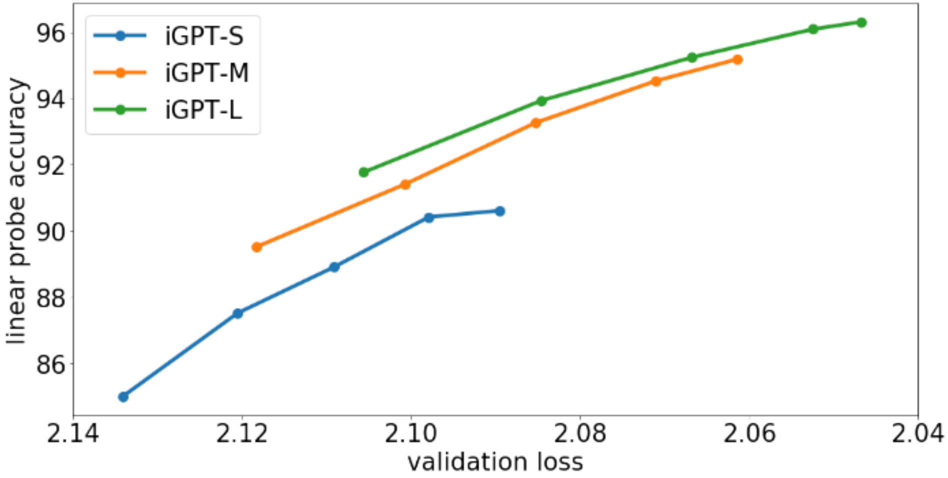
Evaluations for AR

- **Representations at different layers:**
  - Linear probes for the features at different layers
  - Middle layers works the best (Not the last layer?)



Evaluations for AR

- **Better generative models learn better representations:**
  - Dotted markers denote checkpoints at steps 65K, 131K, 262K, 514K, and 1000K



Evaluations for AR

- **Linear probes on CIFAR and STL-10:**
  - Pre-trained on ImageNet
  - Outperform STOA unsupervised pre-training methods

- pre-train with 32x32 down-sampled images
- linear probes with 32x32 images

- pre-train with 224x224 images
- Transfer learning with 224x224 up-sampled images

Model	Acc	Unsup Transfer	Sup Transfer
CIFAR-10			
ResNet-152	94		√
SimCLR	95.3	√	
iGPT-L	96.3	√	
CIFAR-100			
ResNet-152	78.0		√
SimCLR	80.2	√	
iGPT-L	82.8	√	
STL-10			
AMDIM-L	94.2	√	
iGPT-L	95.5	√	

Evaluations for AR

- **Linear probes on ImageNet:**
  - Comparable but not better performance than STOA (at that time)
  - Much larger model and longer features

Method	IR	Params (M)	Features	Acc
Rotation	orig.	86	8192	55.4
iGPT-L	$32^2 \cdot 3$	1362	1536	60.3
BigBiGAN	orig.	86	8192	61.3
iGPT-L	$48^2 \cdot 3$	1362	1536	65.2
AMDIM	orig.	626	8192	68.1
MoCo	orig.	375	8192	68.6
iGPT-XL	$64^2 \cdot 3$	6801	3072	68.7
SimCLR	orig.	24	2048	69.3
CPC v2	orig.	303	8192	71.5
iGPT-XL	$64^2 \cdot 3$	6801	15360	72.0
SimCLR	orig.	375	8192	76.5

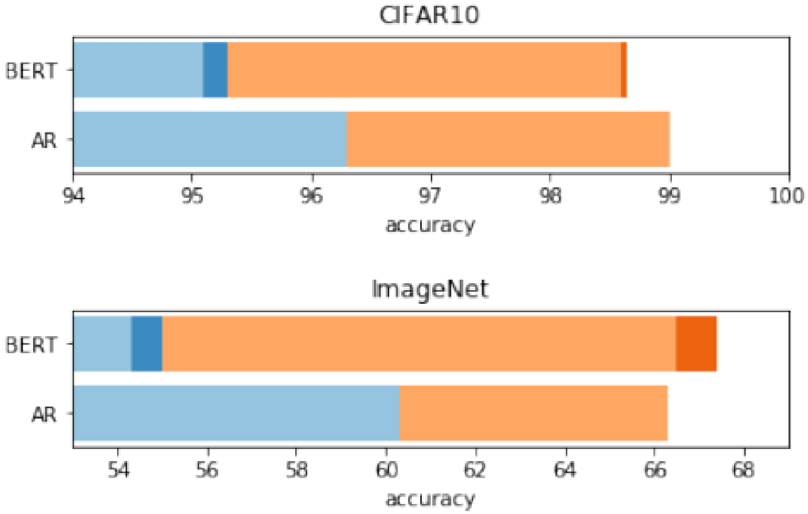
Evaluations for AR

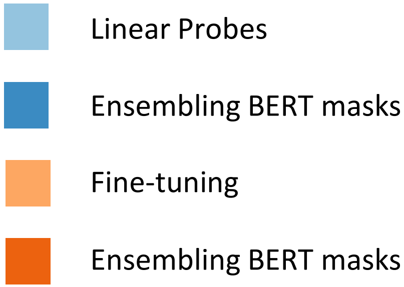
- **Full fine-tuning:**
  - Pre-trained on ImageNet
  - Achieve STOA performance on CIFAR-10
  - Achieve *66.3%* on ImageNet with 32x32x3, which is worse than the STOA performance of *70.2%* (Isometrix Neural Nets)

Model	Acc	Unsup Transfer	Sup Transfer
CIFAR-10			
AutoAugment	98.5		
SimCLR	98.6	√	
GPipe	99.0		√
iGPT-L	99.0	√	
CIFAR-100			
iGPT-L	88.5	√	
SimCLR	89.0	√	
AutoAugment	89.3		
EfficientNet	91.7		√

Evaluations for AR and BERT

- **BERT vs AR:**





- AR outperforms BERT with linear probes
- BERT catches up with fine-tuning

Summary

- **Pros:**
  - Transformers do work for image pre-training
  - Achieves impressive performance on small dataset such as CIFAR-10
  - Large potential for future improvements (e.g., combine with CNN for large images, GPT-3, etc)
- **Cons:**
  - Difficult to deal with high resolution images. Downsampling causes a loss of information
  - Huge memory and computation cost
  - Ignoring the spatial information