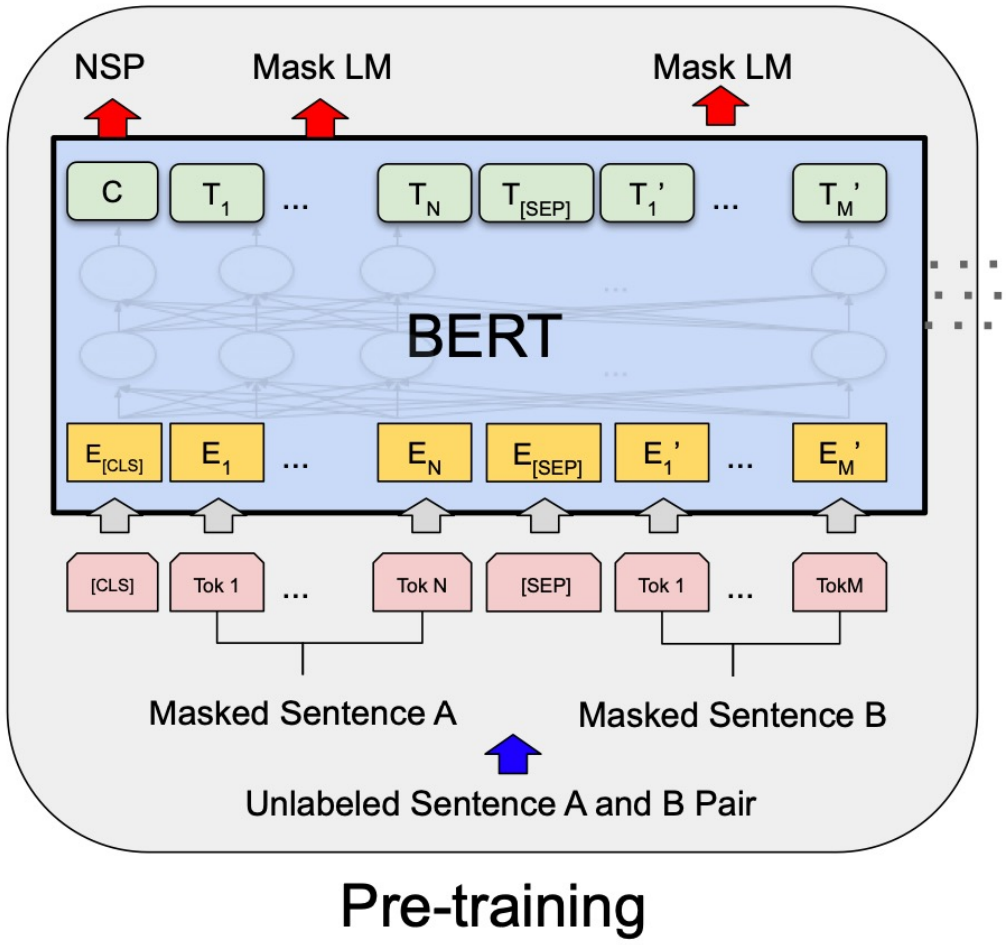
Understanding Masked Pre-training: Fundamentally Different?

Yingzhen Li

yingzhen.li@imperial.ac.uk

What is Masked Pre-Training?

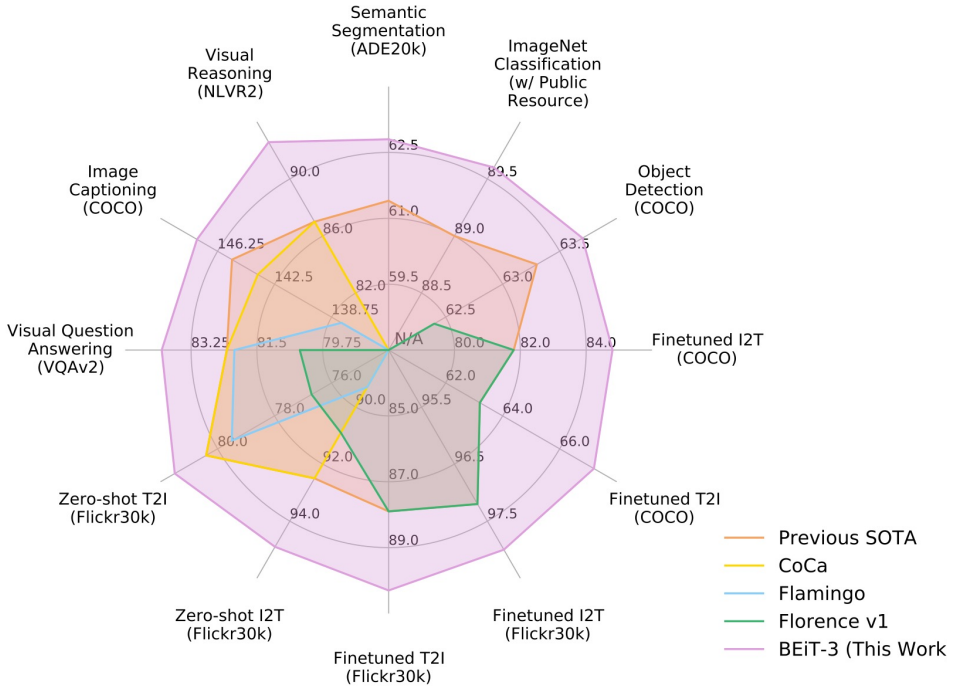
- Train the network to predict the missing words given an incomplete sentence
- During training, randomly masking out some words to create incomplete sentence & prediction targets

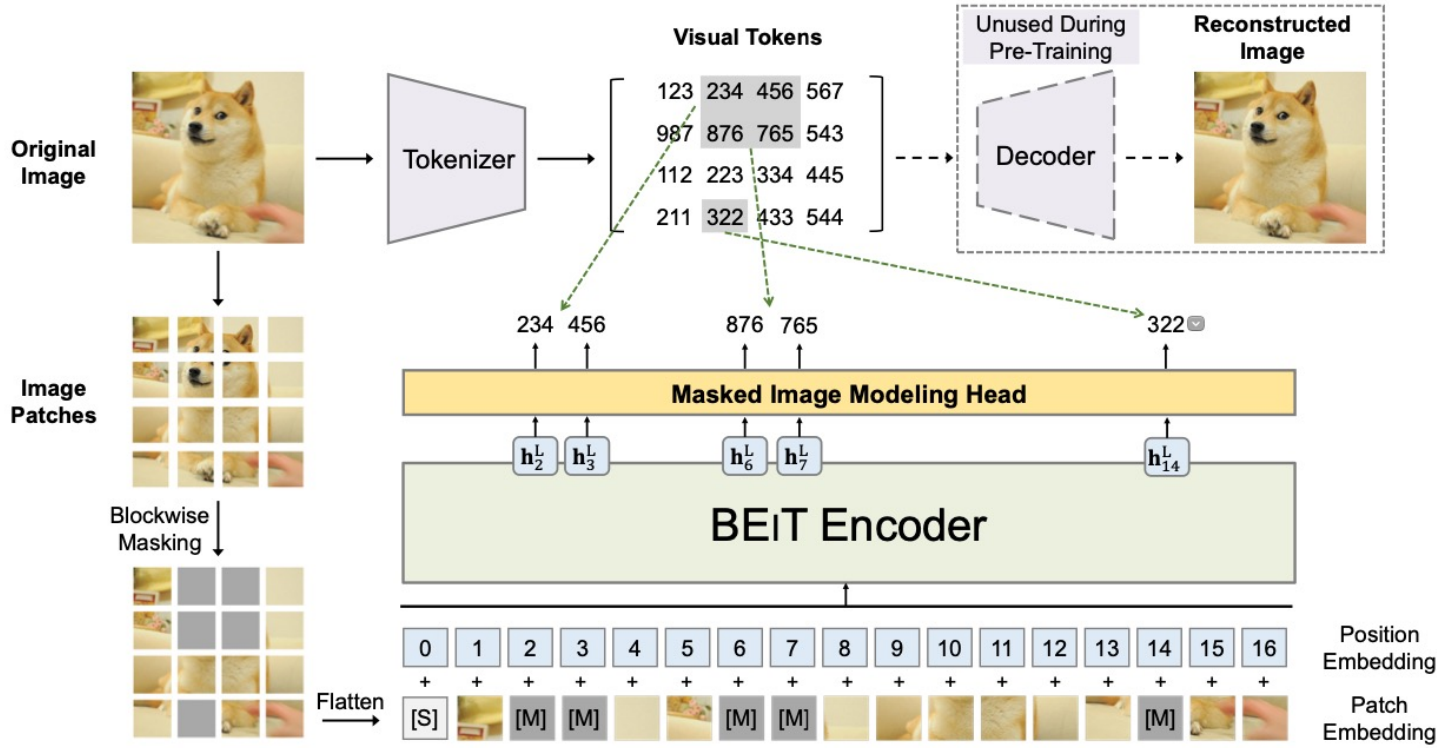


Pre-training

Successes of Masked Pre-Training

SOTA vision-language model (BEiT-3) uses Masked Pre-Training

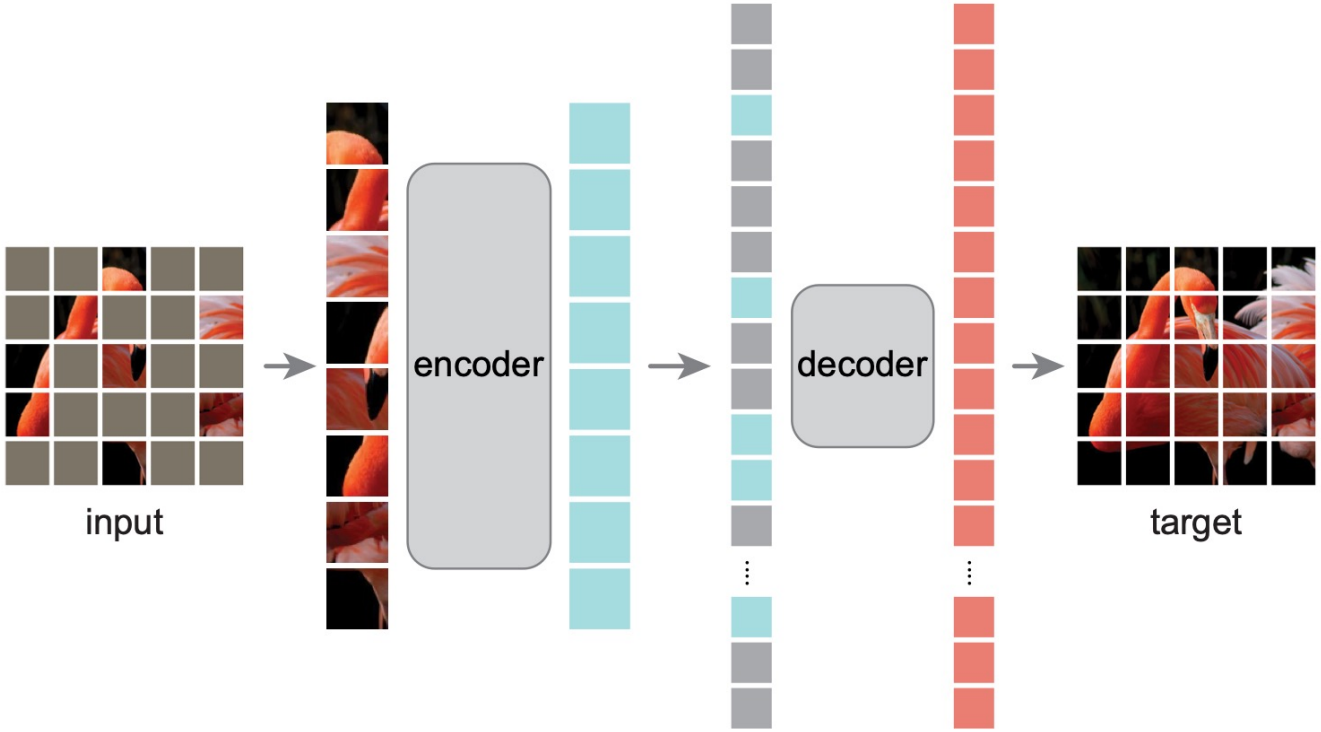


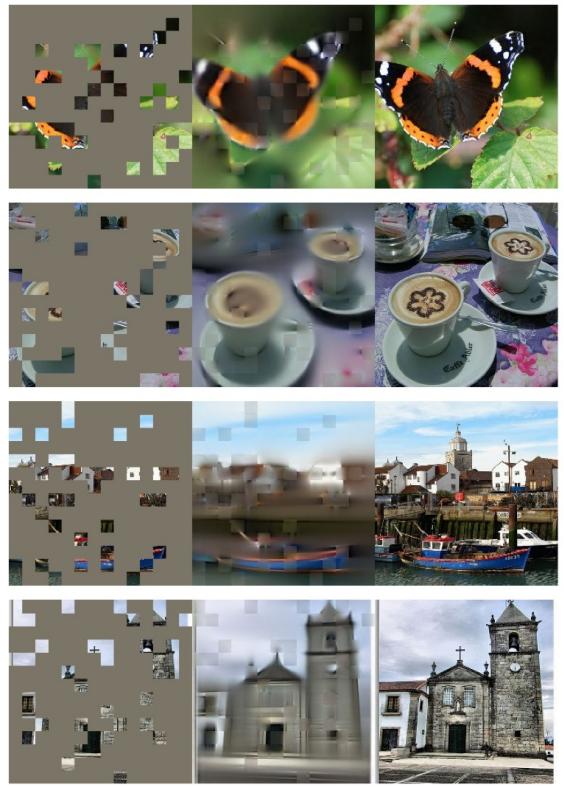


Bao et al. BEiT: BERT Pre-Training of Image Transformers. ICLR 2022
Wang et al. Image as a Foreign Language: BEiT Pretraining for All Vision and Vision-Language Tasks. arXiv:2208.10442

Successes of Masked Pre-Training

Masked auto-encoder for vision: again using Masked Pre-Training



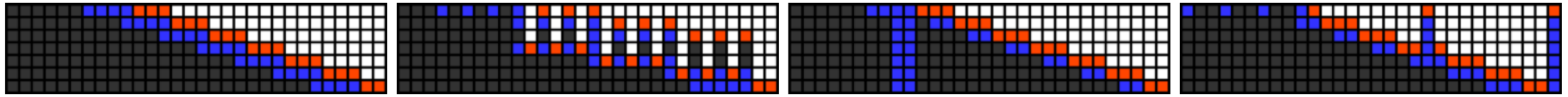


He et al. Masked Autoencoders Are Scalable Vision Learners. CVPR 2022

Successes of Masked Pre-Training

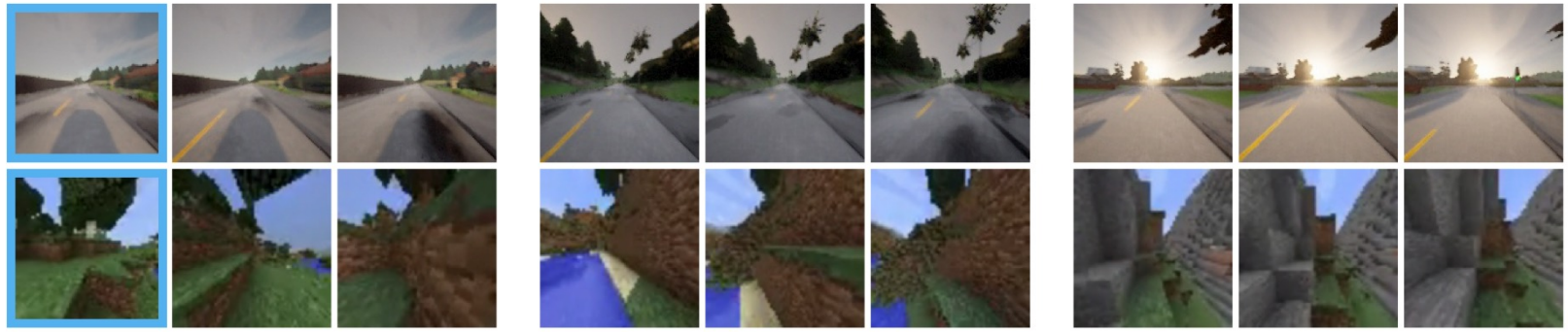
SOTA long-video generation model (up to 15K frames):
Modelling $p(x_P \mid x_C)$ for any prediction time index set P given any conditioning time index set C

Sampling stages



(a) Autoregressive. (b) Two temporal res. (c) Long-range (ours). (d) Hierarchy-2 (ours).

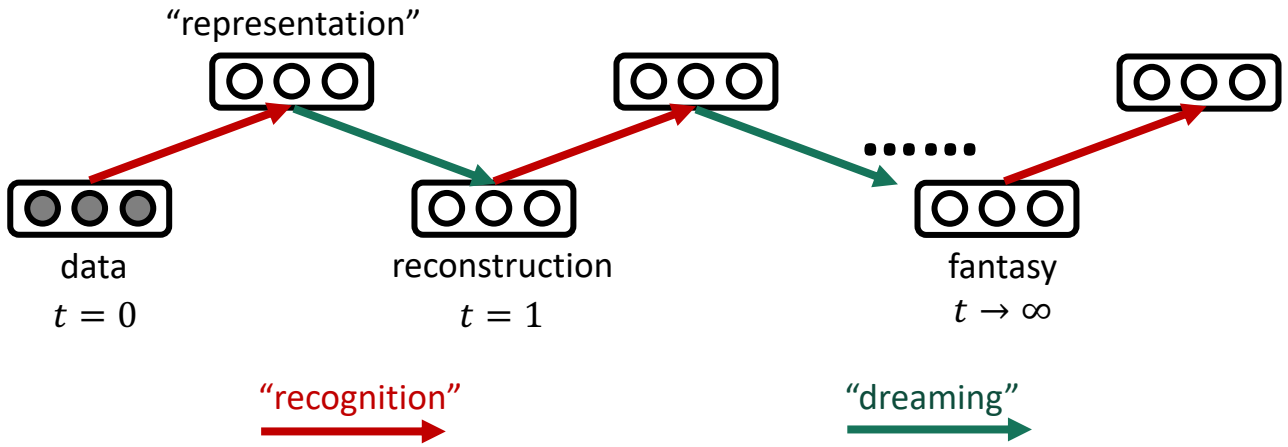
Generation results

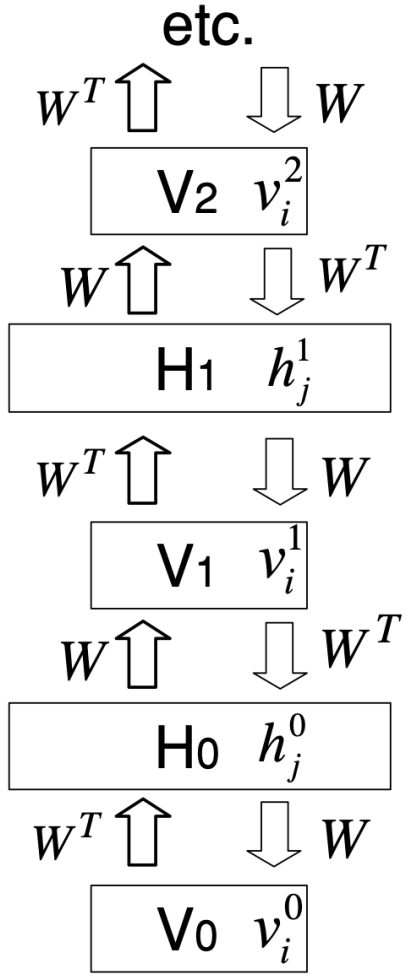


0 5 minutes 25 minutes

“Old Fashioned” Pre-Training

- Deep learning resurgence: pre-training Deep Belief Nets
  - Each layer modelled as a restricted Boltzmann machine
  - Trained using contrastive divergence
  - Layer-wised pre-training





Hinton et al. A fast learning algorithm for deep belief nets[J]. Neural computation, 2006, 18(7): 1527-1554.
Bengio Y. Learning Deep Architectures for AI. Foundations and trends in Machine Learning, 2009, 2(1), 1-127

“Old Fashioned” Pre-Training

ICML 2009

for Scalab

Honglak Lee
Roger Grosse
Rajesh Rangai
Andrew Y. Ng
Computer Scienc

How

Jasc

[3] Dept
[4] I

Multimodal Learning with Deep Boltzmann Machines

Nitish Srivastava NITISH@CS.TORONTO.EDU
Department of Computer Science
University of Toronto
10 Kings College Road, Rm 3302
Toronto, Ontario, M5S 3G4, Canada.

Ruslan Salakhutdinov RSALAKHU@CS.TORONTO.EDU
Department of Statistics and Computer Science
University of Toronto
10 Kings College Road, Rm 3302
Toronto, Ontario, M5S 3G4, Canada.

JMLR 2014

Not in use anymore!

- Not enough compute to make big models
- Not enough data

“Old Fashioned” Unsupervised Learning

- Dimensionality Reduction
  - PCA
  - Auto-encoders
  - …
- Clustering
- Learning a probabilistic generative model (i.e. modelling $p(x)$)
  - VAEs
  - Flows
  - …

Lots of progress has been made here!
… and now we have much more data & compute

A 10M Dollar Question

With $10M GPU computing budget…

Would you try pre-training any big models with any “old-fashioned” unsupervised learning techniques?

Before You Burn the $10M GPU Budget…

- Is masked pre-training fundamentally different?
- Anything BAD about the “old fashioned” methods?

- How to answer these questions?
  - Modelling choices?
  - Advantages from Training objectives?
  - Optimisation aspects?

Modelling Conditional Distributions

Complete Conditionals:

$$\prod_{d=1}^{d_x} p(x_d | x_{-d})$$

Full joint distribution:

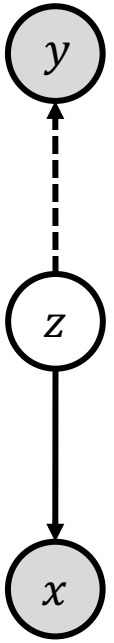
$$p(x_{1:d_x})$$

- Modelling complete conditionals vs full joint distribution:
  - For some models, conditional distributions are much easier to compute
    - Markov random field
    - Restricted Boltzmann machine
  - Related to Gibbs sampling
    - Capturing complete conditionals ⇒ use Gibbs sampling to sample from the joint

Modelling Conditional Distributions

- Complete Conditionals for representation learning (linear case):
  - Assume a linear relationship for the generative process:
    - Continuous case: $p(x|z) = N(Wz, \sigma^2 I)$
    - Discrete case: $p(x = i|z = j) = W_{ij}$

- Capturing the posterior $p(z|x)$ via the complete conditional:
  - The long vector $[p(x_1|x_{-1}), p(x_2|x_{-2}), \ldots, p(x_D|x_{-D})]$ can be represented as a linear transform of $p(z|x)$
    - Under assumptions on W (e.g., full column rank)
  - Learned useful features if the ground truth $x \rightarrow y$ function depends directly on $p(z|x)$
- Similar ideas can be extended to Hidden Markov Models



Wei et al. Why Do Pretrained Language Models Help in Downstream Tasks? An Analysis of Head and Prompt Tuning. NeurIPS 2021
Liu et al. Masked prediction tasks: a parameter identifiability view. arXiv:2202.09305
Luo et al. One Objective for All Models -- Self-supervised Learning for Topic Models. arXiv:2203.03539

Modelling Conditional Distributions

Complete Conditionals:

$$\prod_{d=1}^{d_x} p(x_d | x_{-d})$$

Full joint distribution:

$$p(x_{1:d_x})$$

- Comparing the two modelling paradigms:
  - Capturing information about the posterior $p(z|x)$:
    - Both methods can do (under linear model assumptions)
  - Identifiability of the model
    - Even when the parameter of $p(x_{1:d_x})$ is identifiable, we might not be able to recover that parameter by modelling some conditional distributions

Liu et al. Masked prediction tasks: a parameter identifiability view. arXiv:2202.09305
Gassiat et al. Inference in finite state space non parametric hidden Markov models and applications[J]. Statistics and Computing, 2016, 26(1): 61-71

Modelling Conditional Distributions

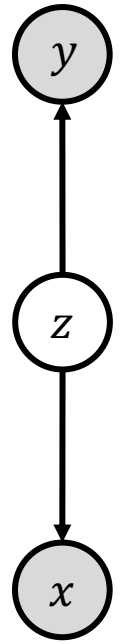
- In practice: we directly parameterise $p(x_d|x_{-d})$
  - Importantly, $p(x_i|x_{-i})$ and $p(x_j|x_{-j})$ share the same network
  - Different from first defining $p(x_{1:d_x})$ then working out $p(x_d|x_{-d})$
- The conditional distributions might not be compatible
  - Circumvented by universal approximations of neural networks?
  - Inconsistency issues: do they matter?

Pre-Training with Masking

- Ground-truth data generation process:
  - $z \sim N(0, \Lambda), z \in R^{d_z}$
  - $x = Wz + \epsilon_x\ , x \in R^{d_x}$
  - $y \sim p(y|z),\ y \in R^{d_y}$
- Regression with linear features: define $f(x) = Bx$, solve

$$\theta^*(B) = argmin\ E_{x,y}[\|y\ - \theta f(x)\|_2^2]$$

  - Specifically we consider $f \in R^{d_f}$ with $d_f < d_x$ (bottleneck)
  - And $d_f < d_z$, i.e., model is mis-specified



Pre-Training with Masking

$z \sim N(0, \Lambda), z \in R^{d_z}$
$x = Wz + \epsilon_x, x \in R^{d_x}$
$y \sim p(y|z), y \in R^{d_y}$

- Regression with linear features: define $f(x) = Bx$, solve
$$\theta^*(B) = argmin\, E_{x,y}[\|y - \theta f(x)\|_2^2]$$
  - Bottleneck mis-specified: $f \in R^{d_f}$ with $d_f < d_x, d_f < d_z$

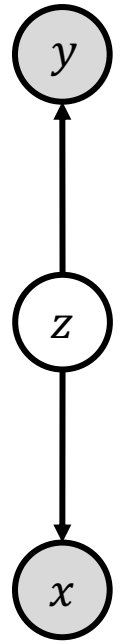
- Find B with linear auto-encoder: solve
$$A^*, B^* = argmin\, E_x[\|x - ABx\|_2^2]$$

$B \in R^{d_f \times d_x}$

Solution: equivalent to PCA up to an invertible matrix

$B^* = C^{-1} U_{d_f}^\top$, U_{d_f} captures the top d_f eigenvectors of Σ_{xx}

$(\Sigma_{xx} = \sigma_x^2 I + W \Lambda W^\top)$

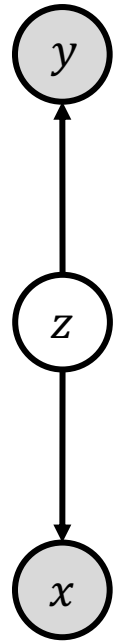


Representation learning works well if y only depends on the subspace of z found by PCA

Baldi and Hornik. Neural networks and principal component analysis: Learning from examples without local minima[J]. Neural networks, 1989, 2(1): 53-58.

Pre-Training with Masking

$z \sim N(0, \Lambda), z \in R^{d_z}$
$x = Wz + \epsilon_x, x \in R^{d_x}$
$y \sim p(y|z), y \in R^{d_y}$
$\Sigma_{xx} = \sigma_x^2 I + W\Lambda W^\top$



- Regression with linear features: define $f(x) = Bx$, solve

$$\theta^*(B) = argmin\ E_{x,y}[\|y - \theta f(x)\|_2^2]$$

  - Bottleneck mis-specified: $f \in R^{d_f}$ with $d_f < d_x, d_f < d_z$

- Find B with masked linear auto-encoder:
  - Define $x_{-j}(e) = [x_1, \ldots, x_{j-1}, e, x_{j+1}, \ldots, x_{d_x}]^\top$
  - Solve the following objective: notice $A \in R^{1 \times d_f}$

$B \in R^{d_f \times d_x}$

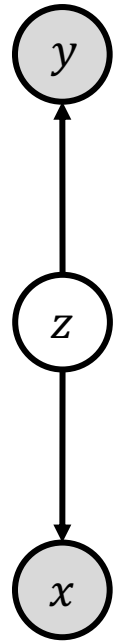
$$A^*, B^* = argmin\ \sum_j E_x[\left(x_j - ABx_{-j}(e)\right)^2]$$

Solution is not unique (when $d_f > 1$), but of the following form:

$$A^* = \left[1, 0_{1 \times d_f - 1}\right] C, B^* = C^{-1}[\begin{matrix} b^* \\ L \end{matrix}], b^* = \underbrace{\left(\Sigma_{xx} 1 - diag(\Sigma_{xx})\right)^\top}_{\sum_j E[x_j x_{-j}(e)^\top]} \underbrace{[(d_x - 1)\Sigma_{xx} + \mathrm{e}^2 \mathrm{I}]^{-1}}_{\sum_j E[x_{-j}(e)\, x_j(e)^\top]}$$

Pre-Training with Masking

$$z \sim N(0, \Lambda), z \in R^{d_z}$$
$$x = Wz + \epsilon_x, x \in R^{d_x}$$
$$y \sim p(y|z), y \in R^{d_y}$$
$$\Sigma_{xx} = \sigma_x^2 I + W\Lambda W^\top$$



- Regression with linear features: define $f(x) = Bx$, solve
$$\theta^*(B) = argmin\, E_{x,y}[\|y - \theta f(x)\|_2^2]$$
  - Bottleneck mis-specified: $f \in R^{d_f}$ with $d_f < d_x$, $d_f < d_z$

- Find B with masked linear auto-encoder:

Arbitrary invertible $d_f \times d_f$ matrix

Mask embedding

$$B^* = C^{-1}\begin{bmatrix} b^* \\ L \end{bmatrix}, \quad b^* = \left(\Sigma_{xx}1 - diag(\Sigma_{xx})\right)^\top [(d_x - 1)\Sigma_{xx} + \mathrm{e}^2\mathrm{I}]^{-1}$$

Arbitrary $(d_f - 1) \times d_x$ matrix, $d_f > 1$

- b^* contains less information about x as compared with the PCA solution
- However, flexibility in L and e allows B to project to different subspaces of z

Pre-Training with Masking

$z \sim N(0, \Lambda), z \in R^{d_z}$
$x = Wz + \epsilon_x, x \in R^{d_x}$
$y \sim p(y|z), y \in R^{d_y}$
$\Sigma_{xx} = \sigma_x^2 I + W\Lambda W^\top$

- Regression with linear features: define $f(x) = Bx$
  - Bottleneck mis-specified: $f \in R^{d_f}$ with $d_f < d_x, d_f < d_z$
- Find B, θ together with supervised learning:

$B \in R^{d_f \times d_x}$

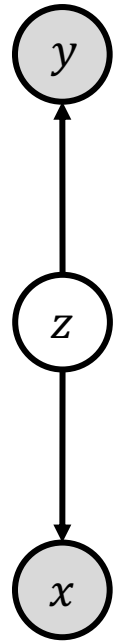
$$\theta^*, B^* = argmin\, E_{x,y}[\|y - \theta B x\|_2^2]$$

- Consider the case of $d_y < d_f < d_z$:

  Solution is not unique, but of the following form:

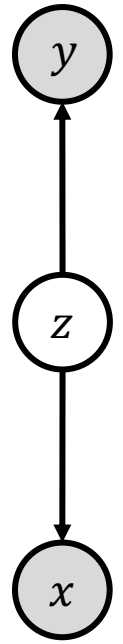
$$\theta^* = \left[I_{d_y \times d_y}, 0_{d_y \times (d_f - d_y)}\right] C, B^* = C^{-1}\begin{bmatrix}\Sigma_{yx}\Sigma_{xx}^{-1} \\ L\end{bmatrix},$$

($\theta^* B^* = \Sigma_{yx}\Sigma_{xx}^{-1}$ is the least squares solution of linear regression)



Pre-Training with Masking

$z \sim N(0, \Lambda), z \in R^{d_z}$
$x = Wz + \epsilon_x, x \in R^{d_x}$
$y \sim p(y|z), y \in R^{d_y}$
$\Sigma_{xx} = \sigma_x^2 I + W\Lambda W^\top$



- Regression with linear features: define $f(x) = Bx$
  - Bottleneck mis-specified: $f \in R^{d_f}$ with $d_f < d_x, d_f < d_z$
- Now compare the two solutions: assume $d_y < d_f < d_z$

Supervised learning:

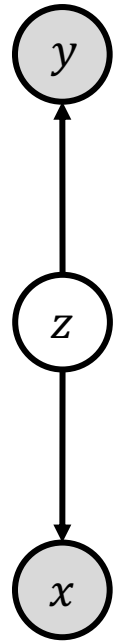
$$B^* = C_s^{-1}\begin{bmatrix}\Sigma_{yx}\Sigma_{xx}^{-1}\\ L_s\end{bmatrix},$$

Masked linear auto-encoder:

$$B^* = C_m^{-1}\begin{bmatrix}b^*\\ L_m\end{bmatrix}, \quad b^* = \left(\Sigma_{xx}1 - diag(\Sigma_{xx})\right)^\top [(d_x - 1)\Sigma_{xx} + \mathrm{e}^2\mathrm{I}]^{-1}$$

Pre-Training with Masking

$z \sim N(0, \Lambda), z \in R^{d_z}$
$x = Wz + \epsilon_x, x \in R^{d_x}$
$y \sim p(y|z), y \in R^{d_y}$
$\Sigma_{xx} = \sigma_x^2 I + W\Lambda W^\top$



- Regression with linear features: define $f(x) = Bx$
  - Bottleneck mis-specified: $f \in R^{d_f}$ with $d_f < d_x, d_f < d_z$
- Now compare the two solutions: assume $d_y < d_f < d_z$

Supervised learning:

Arbitrary invertible $d_f \times d_f$ matrix

$$B^* = C_s^{-1}\begin{bmatrix}\Sigma_{yx}\Sigma_{xx}^{-1} \\ L_s\end{bmatrix},$$

Arbitrary $(d_f - d_y) \times d_x$ matrix

Masked linear auto-encoder: (setting $e = 0$)

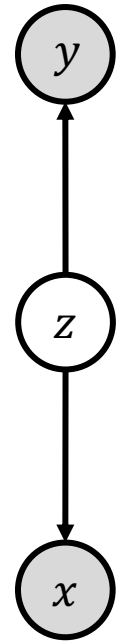
$$B^* = C_m^{-1}\begin{bmatrix}b^* \\ L_m\end{bmatrix}, \quad b^* = \frac{1}{d_x - 1}\left(\Sigma_{xx}1 - diag(\Sigma_{xx})\right)^\top \Sigma_{xx}^{-1}$$

Arbitrary invertible $d_f \times d_f$ matrix

Arbitrary $(d_f - 1) \times d_x$ matrix

Pre-Training with Masking

$z \sim N(0, \Lambda), z \in R^{d_z}$
$x = Wz + \epsilon_x, x \in R^{d_x}$
$y \sim p(y|z), y \in R^{d_y}$
$\Sigma_{xx} = \sigma_x^2 I + W\Lambda W^\top$



- Regression with linear features: define $f(x) = Bx$
  - Bottleneck mis-specified: $f \in R^{d_f}$ with $d_f < d_x, d_f < d_z$
- Now compare the two solutions: assume $d_y < d_f < d_z$

Making the two solutions equal: find matrices C and L such that

$$\underbrace{\frac{1}{d_x - 1}\left(\Sigma_{xx}1 - diag(\Sigma_{xx})\right)^\top}_{\in R^{1\times d_x}} = C_{1\cdot}\left[\begin{matrix}\Sigma_{yx} \\ L\end{matrix}\right],$$

$C_{1\cdot} \in R^{1\times d_f}$ (the 1st row of C)

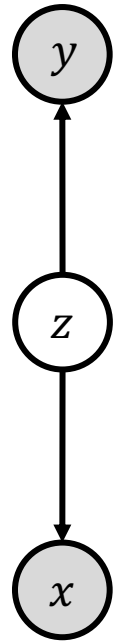
$L \in R^{(d_f - d_y)\times d_x}$

"solving a set of d_x equations with $d_f + (d_f - d_y)\times d_x$ variables"

$d_y < d_f \Rightarrow d_f - d_y \geq 1 \Rightarrow d_f + (d_f - d_y)\times d_x > d_x$ ⇒ solution exists!

Pre-Training with Masking

$z \sim N(0, \Lambda), z \in R^{d_z}$
$x = Wz + \epsilon_x, x \in R^{d_x}$
$y \sim p(y|z), y \in R^{d_y}$
$\Sigma_{xx} = \sigma_x^2 I + W \Lambda W^\top$

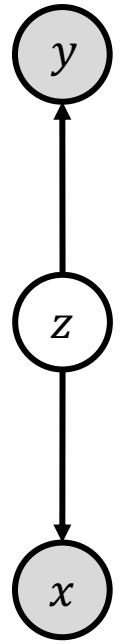


- Regression with linear features: define $f(x) = Bx$
  - Bottleneck mis-specified: $f \in R^{d_f}$ with $d_f < d_x, d_f < d_z$
- Compare - w/out masking :
  - Linear auto-encoder:
    - Solution equivalent to PCA
    - Best linear representation for x (in terms of ℓ_2 recon. error)
    - Might not be good for prediction (depending on ground truth $p(y|z)$)
  - Masked linear auto-encoder:
    - Solution is not unique (when $d_f > 1$)
    - Not so good representation for x (in terms of ℓ_2 recon. error)
    - A subset of solutions are optimal for prediction (when $d_y < d_f < d_z$)

Hypothesis: masked pre-training works better when good representation for generating x is highly redundant

Pre-Training with Masking

$z \sim N(0, \Lambda), z \in R^{d_z}$
$x = Wz + \epsilon_x, x \in R^{d_x}$
$y \sim p(y|z), y \in R^{d_y}$
$\Sigma_{xx} = \sigma_x^2 I + W\Lambda W^\top$



- Regression with linear features: define $f(x) = Bx$
  - Bottleneck mis-specified: $f \in R^{d_f}$ with $d_f < d_x, d_f < d_z$
- Caveats of this example:
  - Masked linear auto-encoder:
    - A subset of solutions are optimal for prediction (when $d_y < d_f < d_z$)
    - But: pre-training on x has no obvious "incentives" to find them (optimal B^* in supervised learning depends on Σ_{yx})
  - Extension to non-linear models:
    - With over-parameterised NNs, auto-encoder solution might not be unique
    - But Masked auto-encoder might have bigger subspace of optimal solution for prediction
    - Again: pre-training on x has no obvious "incentives" to find them

Optimisation Aspects

- Lower dimensional target to match:
  - Linear Auto encoder (PCA) features: $f(x) = Bx$

$$A^*, B^* = argmin\ E_x[\|x\ - ABx\|_2^2] \qquad A \in R^{d_x \times d_f}$$

  - Masked Linear Auto encoder features: $f(x) = Bx$

$$A^*, B^* = argmin\ \sum_j E_x[\left(x_j - ABx_{\_j}(e)\right)^2] \qquad A \in R^{1 \times d_f}$$

Optimisation Aspects

- More equally good optima (for representation learning)?
  - Linear Auto encoder (PCA) features: $f(x) = Bx$

Arbitrary invertible matrix

$$B^* = C^{-1}U_{d_f}^\top, \qquad U = eig(\Sigma_{xx})$$

  - Masked Linear Auto encoder features: $f(x) = Bx$

Arbitrary invertible matrix

Mask embedding

$$B^* = C^{-1}\begin{bmatrix} b^* \\ L \end{bmatrix}, \quad b^* = \left(\Sigma_{xx}1 - diag(\Sigma_{xx})\right)^\top [(d_x - 1)\Sigma_{xx} + \mathrm{e}^2\mathrm{I}]^{-1}$$

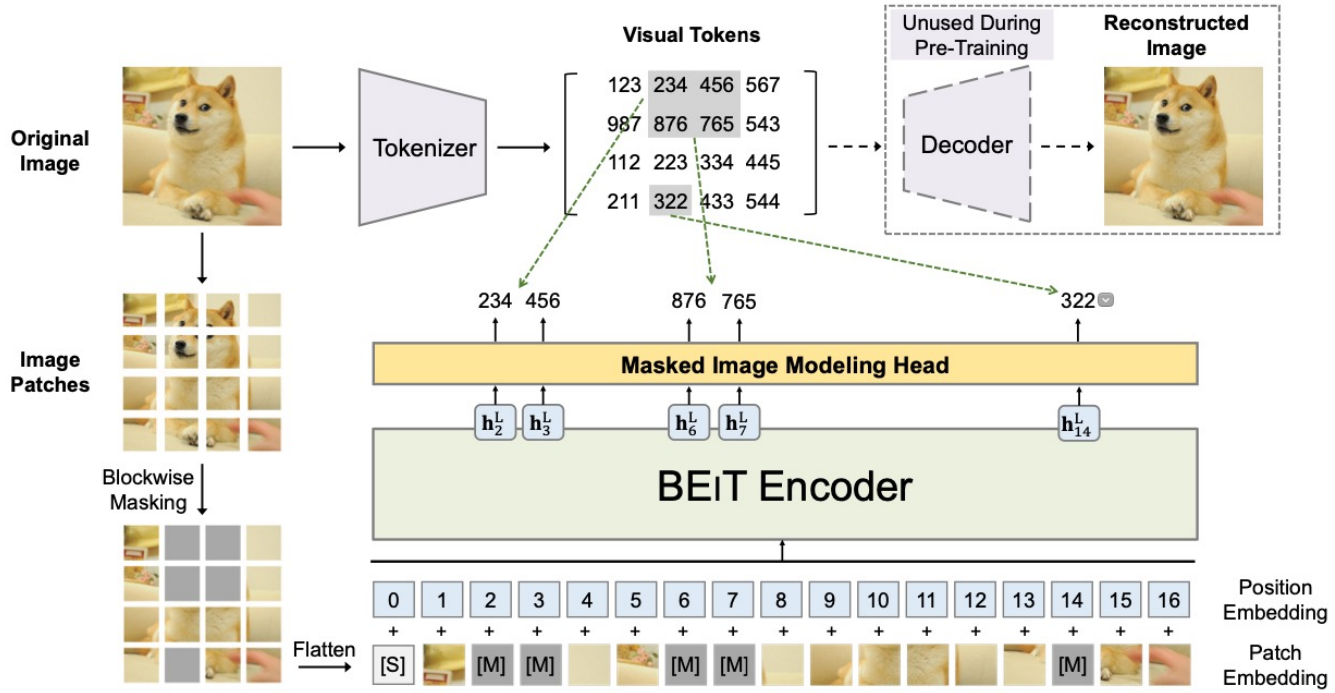
Arbitrary $(d_f - 1)\times d_x$ matrix

Optimisation Aspects

- Current DL tricks engineered towards supervised learning tasks?

BEiT as an example:

- Pretraining task: classification
- (sparse) Cross-entropy loss



Bao et al. BEiT: BERT Pre-Training of Image Transformers. ICLR 2022
Wang et al. Image as a Foreign Language: BEiT Pretraining for All Vision and Vision-Language Tasks. arXiv:2208.10442

Still Consider Burning the $10M GPU Budget?

- Is masked pre-training fundamentally different?
- How to answer this question?
  - Modelling choices?
    - Modelling Conditional distributions: a better choice?
    - Not the full story: incompatibility/inconsistency issue matters or not?
  - Advantages from Training objectives?
    - More flexible representation learning?
    - Redundancy in information of x?
    - Any benefit of the extra [mask] embedding e?
  - Optimisation aspects?
    - Easier for optimisation? due to lower-dim targets (many equally good optima)?
    - Deep neural networks designed to benefit supervised learning objectives?
  - Other perspectives (from you)?